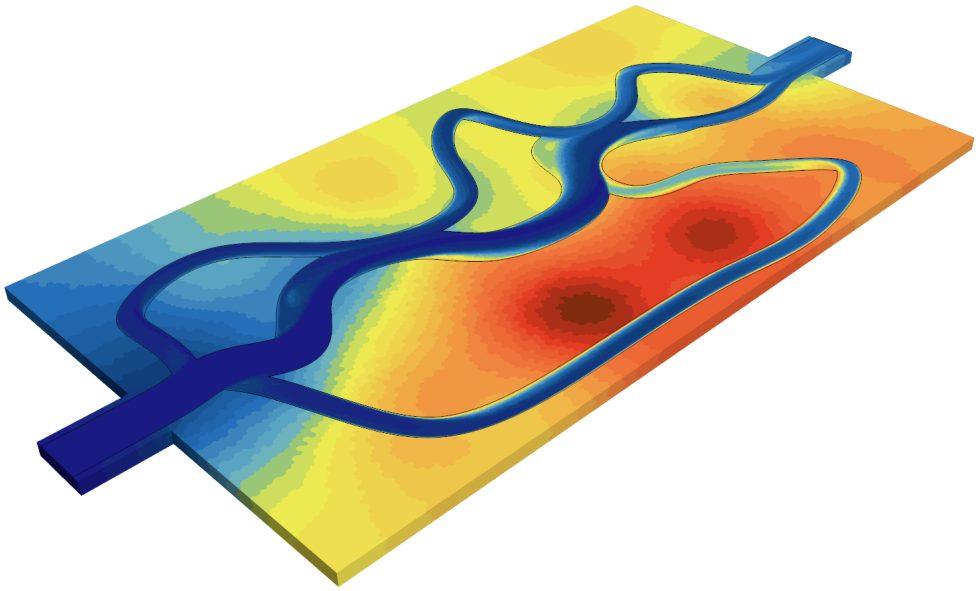
CHALMERS



Machine Learning-Based Optimization for Battery Pack Cooling

TME180 Automotive Engineering Project 2024

Abubakar Abukar
Pedro John
Francisco Boudagh
Salvatore Verde
Gabriel Wendel

Department of Mechanics and Maritime Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2024

Machine Learning-Based Optimization for Battery Pack Cooling

TME180 Automotive Engineering Project 2024

Supervisor: Alexey Vdovin, Department of Mechanics and Maritime Sciences
Supervisor: Anthony Vivek, AFRY
Supervisor: Dimitrios Koutsimanis, AFRY
Supervisor: Viktor Alatalo, AFRY
Examiner: Alexey Vdovin, Department of Mechanics and Maritime Sciences

Studentarbeten – Mekanik och maritima vetenskaper (M2) – Projektarbete
Department of Mechanics and Maritime Sciences
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone +46 (0)31 772 1000

Abstract

The thermal management of electric vehicle (EV) batteries is a critical factor in safety, performance and durability. Traditional design methods for cooling channels rely on manual adjustments, trial and error processes and intensive topology optimizations leading to time consuming approaches and significant limitations in efficiency. This study introduces an innovative framework that combines machine learning (ML) and computational fluid dynamics (CFD) to optimize cooling channel designs that circumvent the challenges faced by traditional methods. These improvements are achieved by using two different components:

The first component is a surrogate model. It is a machine learning model trained on a large dataset produced through CFD simulations using Star-CCM+. This model significantly reduces computational costs and time by predicting pressure drops and temperature distributions based on input geometries and system parameters.

The second component is a genetic algorithm for geometry optimization. This component generates an optimal geometry by balancing pressure drop and an effective thermal regulation by using a Non-Dominated Sorting Genetic Algorithm (NSGA-II). This algorithm creates a number of random solutions and iteratively improves on them to find a Pareto front, which represents the optimal trade-offs between competing objectives: minimizing pressure drop and maximizing thermal regulation. The role of the surrogate model is to provide instant feedback on potential solutions throughout the algorithm, which is vital as the algorithm itself is inherently time consuming.

By combining these two components, the framework accelerates the design process ensuring the creation of advanced cooling solutions. The results demonstrate the potential to reduce computational time while achieving superior performance. This work provides a framework for future advancements in the thermal management of EV batteries, highlighting the importance of combining CFD with ML in modern engineering solutions.

Contents

Acronyms

CFD Computational Fluid Dynamics.

EA Evolutionary Algorithm.

FNO Fourier Neural Operator.

LIB Lithium-Ion Battery.

MAE Mean Absolute Error.

ML Machine Learning.

MOTO Multi-Objective Topology Optimization.

MSE Mean Squared Error.

PDE Partial Differential Equation.

SM Surrogate Model.

1 Introduction

Efficient thermal management in electric vehicle (EV) batteries are critical for ensuring optimal performance, safety, and longevity. The geometry of cooling channels between battery cells and other components plays a vital role in maintaining stable temperature levels. However, designing these channels is often achieved through experience-based methods, trial-and-error processes, or computationally intensive topology optimization. While these approaches can yield functional designs, they are time-consuming, require specialized expertise, and heavily depend on computational resources. Additionally, since these methods lack robust optimization frameworks, the resulting designs are rarely globally optimized, leaving large portions of the design space unexplored.

Tailored designs, which adapt cooling channels to specific components and inhomogeneous heat sources, are especially challenging due to the computational cost of creating unique solutions. Instead, generic designs are often produced, sacrificing efficiency for feasibility. Tailored solutions, however, could significantly enhance efficiency by addressing the unique thermal requirements of different components.

Traditional methods leave significant room for improvement. Optimization strategies such as genetic algorithms and other stochastic methods provide promising alternatives for cooling channel design. For example, topology optimization directly coupled with computational fluid dynamics (CFD), as demonstrated by Wu using Star-CCM+ [1], has shown success. However, these methods face a common bottleneck: high computational costs. Each potential solution must be iteratively evaluated using CFD simulations, with a single simulation taking approximately one hour on an 18-core system, equivalent to about 53,000 CPU seconds. Despite these challenges, CFD remains the most reliable and accurate tool for simulating and analyzing thermal performance.

Recent advancements in machine learning (ML) offer a solution to this computational bottleneck. By training neural networks to predict key metrics, such as heat distribution and pressure drop, significantly reduces evaluation times. A single prediction can be completed in less than a second, and multiple predictions can be performed in parallel within a few seconds. However, the primary challenge lies in ensuring the accuracy of ML predictions. This requires a robust neural network architecture and a carefully curated dataset.

This study proposes a framework for generating such datasets and developing an ML architecture that ensures accurate predictions. By addressing the computational bottleneck, this framework enables the use of optimization models to create cooling channel designs that balance thermal efficiency and pressure drop.

1.1 Aim

The aim of this study to investigate innovative Machine Learning (ML) approaches to streamline and enhance the design process of cooling channels for different heat configurations. Our approach is divided into two components:

- Surrogate Model (SM): An ML model will be trained to predict the pressure drop and temperature distribution of a system based on a given cooling channel geometry, heat source, and inlet velocity. To accomplish this, comprehensive datasets will be generated using computational fluid dynamics (CFD) simulations conducted with Simcenter STAR-CCM+.

- Geometry Optimization: Machine Learning (ML) will be employed in order to create an optimal design. The objective is to develop a framework that, given a representation of the heat source, can generate unique cooling channel designs, optimized to minimize

pressure drop and temperature. The algorithm leverages the surrogate model to iteratively assess and optimize proposed geometries, targeting minimal pressure drop and optimal temperature regulation.

By integrating these components, our approach offers a more efficient alternative to traditional design methods. This not only accelerates the cooling channel development process but also facilitates the creation of innovative geometries tailored to specific battery configurations, opening new avenues for enhanced thermal management in electric vehicle batteries.

2 Literature Review

This section explores existing research that forms the foundation for this study, focusing on two main areas: computational fluid dynamics simulations and machine learning.

2.1 Simulations

In order to find the best approach, it is essential to build the base upon existing research on the topic of combination of CFD and ML. Numerous approaches have been explored to enhance the cooling efficiency of cold plates, which is critical in maintaining safe operating temperatures for lithium-ion batteries (LIB). One recent study by Wu used Multi-Objective Topology Optimization (MOTO) on cold plates with branched and streamlined mini-channel design to improve thermal management while at the same time minimizing the energy demand [1]. Wu's study was chosen to benchmark CFD simulations and the same dimensions of the cooling plate presented in the paper were used. The geometric model from the paper served as a good baseline for testing the performance of our simulations, ensuring that the models accurately captured temperature, pressure, and velocity distributions.

2.2 Machine Learning

There are various approaches to how the architecture of the ML models can be designed, each with different trade-offs and strengths. A study by Chen et al. highlights the necessity and difficulty of minimizing the allocated computational power to CFD simulations through the use of ML models [2]. The paper investigates the issue of training ML models to infer temperature distribution for different heat source layouts. The study evaluates and benchmarks the performance of different types of networks, that all utilize convolutional layers, for solving different problems. For this type of problem the main approach often involves using deep neural networks with convolutional layers for its strong performance in learning spatial dependencies. By presenting comparative results, the paper provides a robust analysis of the trade-offs involved in model accuracy, computational efficiency and generalizability.

Fourier Neural Operators (FNOs) are a type of neural network designed to solve problems governed by partial differential equations (PDEs). In contrast to traditional neural networks, FNOs are designed to learn how to map input to desired output through continuous functions (operators). It has been shown that they are effective for tasks like simulating fluid dynamics or heat transfer [3, 4]. Li et al., who first introduced FNOs, showed that FNOs can significantly speed up computations while maintaining high accuracy [5]. In the paper, they demonstrated the ability of FNOs to handle complex tasks through modeling turbulent flows.

Kovachki et al. later showed that FNOs are flexible and can adapt to different PDE problems, even with limited training data [6]. They perform well in tasks such as weather prediction, heat transfer, and fluid dynamics simulations. These studies highlight FNOs as an efficient and scalable alternative to traditional PDE solvers for computationally expensive simulations in physics and engineering.

2.3 Geometry Optimization

One of the algorithms explored for the purpose of generating geometries is stochastic optimization method known as Evolutionary Algorithm (EA). This algorithm attempts to find the optimal solution to a problem by simulating an evolutionary process in which the different solutions serve as individuals in a population. Through this process the solution, individuals, undergo steps like evaluation, selection, mutation and crossover. The solutions are initialized randomly and slowly changed based on their performance with the aforementioned steps. A very important aspect of this algorithm is the evaluation as it controls the overall flow of the algorithm. The correct evaluation implementation is especially vital for multi-objective fitness functions. The classical approach to this issue is to use a weighted sum to balance the objectives. The issue with this approach is that it necessitates trial and error to find weights that lead to the optimal solution. Deb et. al. introduce a novel alternative to the classic approach called the Non-Sorting Genetic Algorithm II (NSGA-II) [7]. This algorithm uses crowding distance in combination with the fitness values in order to optimally explore the search space of solution. The various solutions found by the algorithm are then used to find a set of non-dominated solutions in order to simulate a Pareto front.

3 Methodology

The following chapter describes the approaches used in the project, including the creation of CAD geometries and the simulation set-up in Star-CCM+. It presents the models, boundary conditions and the different meshes used, as well as the methods for recording the data after simulation. Lastly, the chapter provides a detailed explanation of the network architecture for both the surrogate model and the geometry generation model. See Appendix A for all relevant code.

3.1 Geometry Generation

As previously mentioned in Chapter 2, the same dimensions for the cooling plate as those in Wu's paper were used [1], see Figure 1. Different cooling configurations were then created by designing different cooling channel patterns on top of the plate.

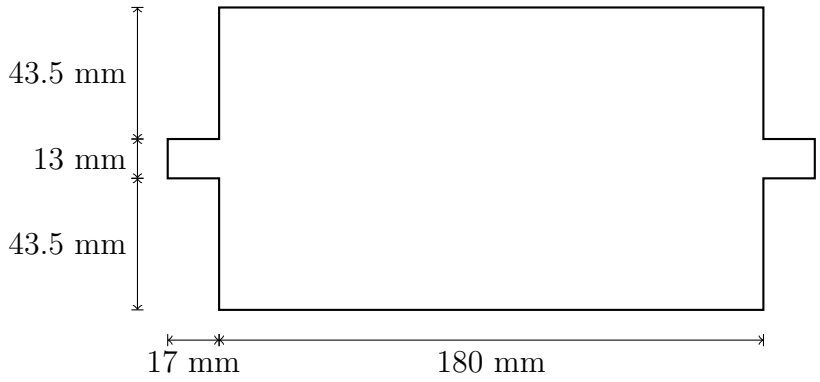


Figure 1: Dimensions of the cooling plate with one inlet and one outlet.

To create a large set of CAD models representing both random and structured cooling configurations, thereby covering a wide spectrum of channel designs, both handmade- and parametric generated CAD models were used for simulation.

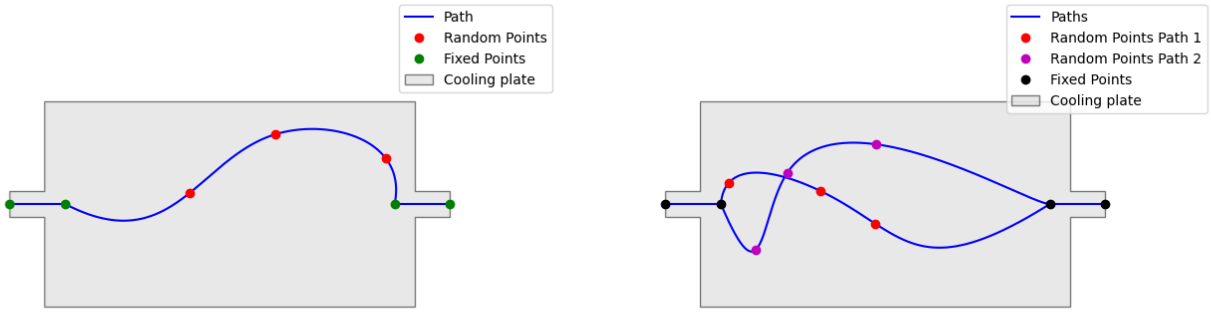
3.1.1 CAD Parametric Modeling

Creating a large number of models by hand is very time-consuming. To address this issue, parametric modeling is implemented. Parametric modeling refers to the process of generating CAD models where the dimensions and features are defined by parameters. The parameters can be modified at any point during the modeling process, and the CAD model automatically updates according to these changes.

Parametric modeling offers both great efficiency and flexibility. Once the base script is written, it can easily be altered or looped to create multiple models in a relatively short amount of time.

Parametric modeling is implemented using Python scripting with FreeCAD. FreeCAD is a 3D modeling software that provides strong capabilities for parametric modeling using its Python API. For this project, a script that automates the creation of random cooling configurations is developed.

The script generates a solid cooling plate based on predefined data points corresponding to the plate's dimensions. A simple algorithm generates paths on the cooling plate, which are used as a reference for creating the cooling channels and the fluid geometry. The algorithm generates random points within the plate boundary and connects them using spline curves. Figure 2 illustrates how the algorithm works for one and two generated paths. To ensure that the paths are straight at the inlet and outlet, fixed points are also defined.



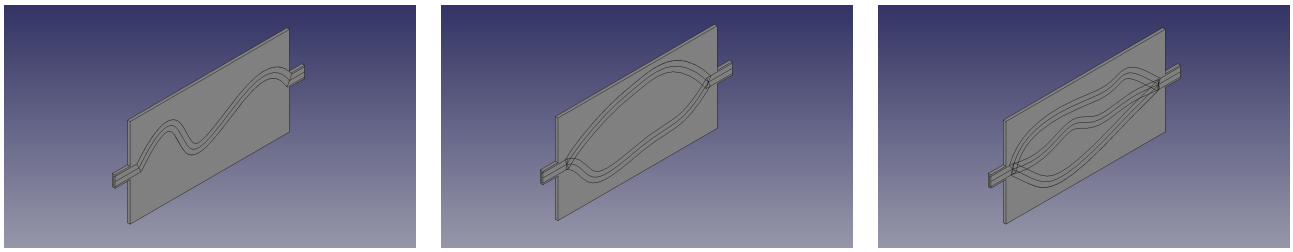
(a) Cooling plate with one generated path.

(b) Cooling plate with two generated paths.

Figure 2: Examples of generated paths.

The number of paths is a parameter that can be altered. To ensure that the points are generated within the plate, a margin to the plate boundaries is defined. The number of random points is also an adjustable parameter. If there is more than one cooling channel, a margin between the channels can also be defined. This ensures that the channels are evenly distributed across the cooling plate, thereby covering a larger portion of its surface.

To create the fluid geometry, the sweep operation is applied to each random path. The swept object is then used in a boolean difference modifier to remove the portion of the fluid object in contact with the solid plate, thereby carving out the plate to form cooling channels. The entire process is placed in a loop that generates the desired number of models and automatically exports them as step files. All files are then imported into Star-CCM+ for simulation, which forms a part of the training data for the surrogate model. Figure 3 illustrates examples of CAD parts generated in FreeCAD using the algorithm. In total 150 geometries were generated and exported for simulation (50 one-, 50 two- and 50 three-channel geometries), in addition to 150 hand made ones.



(a) Plate with one channel.

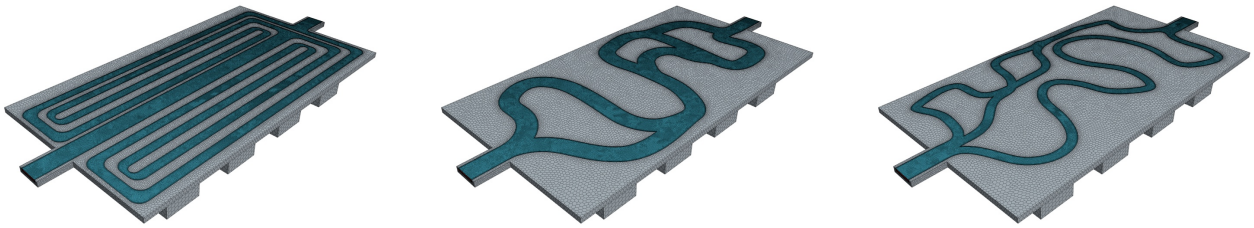
(b) Plate with two channels.

(c) Plate with three channels.

Figure 3: Examples of generated geometries with varying number of channels.

3.1.2 Handmade Channel Geometries

In addition to the algorithmically generated channel geometries, a set of handmade geometries was also created using Siemens NX. This approach allowed for the creation of geometries with a more structured configuration, in contrast to the more random configurations created by algorithmic generation. Figure 4 shows examples of handmade geometries.



(a) Symmetric cooling channel. (b) Asymmetrical with wide path. (c) Asymmetrical with narrow path.

Figure 4: Examples of handmade geometries.

3.2 Simulation Process

The geometries, both hand-made and generated, are imported into the CFD software Star-CCM+ and simulated for different fluid velocities and heat configurations. In order to run a large number of geometries with different heat sources and inlet velocities, a Java script is developed.

The Java script starts from a predefined template and iterates through different configurations. The script firstly takes the channel geometry with its solid and imports it into Star-CCM+. Then it uses specific filters in order to consistently have the same faces which will be used for the regions or the mesh refinements. After that, it selects the inlet velocity along with heat source and runs the simulation. The script selects all the possible configurations of velocity and heat source and once it is finished it starts again from the beginning and selects the neext geometry and solid. The script runs 16 different configurations before it changes the geometry. These are four different inlet velocities (0.1 m/s, 0.2 m/s 0.3 m/s 0.4 m/s) and four different heat source layouts that will be discussed later on.

3.2.1 CFD-Validation

To ensure reliability, the model is validated by comparing Wu's non-optimized study [1]. The total length for the fluid and plate (including the inlet and outlet) is 214 mm and the width is 100 mm. The height of the fluid is 1.5 mm while the height of the solid where the fluid is not present is 3 mm. It has to be considered that the fluid and plate are cut in half for the simulations therefore the height is the double of the stated dimensions. The geometry of the validation has 7 parallel channels that flow from the inlet into the direction of the outlet. The LIB does not have the the inlet and outlet section and is 180 mm long and 100 mm wide with a height of 35 mm. Detailed geometry dimensions are displayed in Figure 5 while Table 1 shows the values that will be compared.

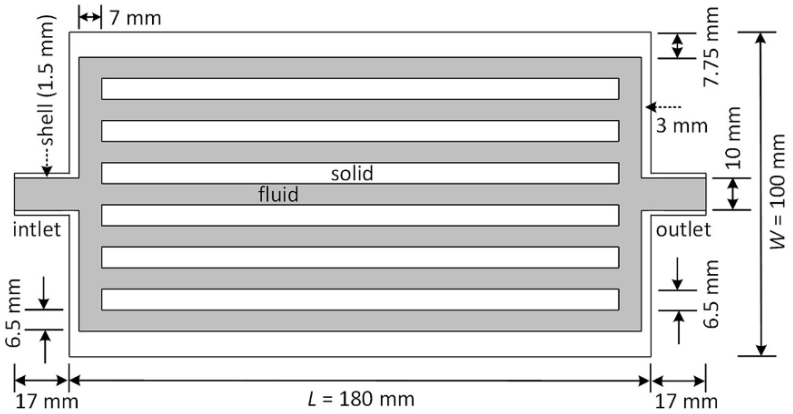


Figure 5: Geometry dimensions [1]

Table 1: Literature values to be compared.

Values	**Literature values**
T_average	74.51 °C
Delta_T	56.86 °C
Delta_P	32.02 Pa

The values that are compared are T_average, Delta T and Delta P. T_average is the average temperature of the geometry, Delta_T is the difference between the max and min temperature in the LIB while Delta_P is the pressure drop between the inlet and the outlet.

For the simulation three physics continua are used. Two for the solids, cooling plate and LIBs, and one for the fluid. The continua for the solids are settled to have a constant density with a coupled solid energy while the fluid was defined as a liquid with a constant density, coupled energy and an All y+ Wall treatment. Here water was selected as the cooling fluid, while aluminum was chosen as the material for the plate. Table 2 has all the Material Properties for the two solids and the fluid listed.

Table 2: Material Properties.

Properties	**Plate**	**LIB**	**Fluid**
Density [kg/m^3]	2719.0	2475.0	998.2
Dynamic Viscosity [Pa − s]	-	-	0.001
Specific Heat [J/kg − K]	871.0	1005.91	4182.0
Thermal Conductivity [W/m − K]	202.4	$k_{xx} = 2.06$ $k_{yy} = 169.42$ $k_{zz} = 169.42$	0.6
Turbulent Prandtl Number	-	-	0.9

The fluid domain has to be a turbulence model, since it will simulate different geometries with also complex geometries and therefore the Reynolds-number will increase. To decide which turbulence model is best, a study is conducted and the following Table 3 shows their performance. T_average and Delta_T where on nearly the same value as the literature values and therefore only delta_P was confronted.

Table 3: Comparison turbulence models

Turbulence model	**Delta_P**
Spalart-Allmaras	33,1 Pa
Standard two layer k-Epsilon	29,6 Pa
Realizable two layer k-Epsilon	29,7 Pa
Elliptic Blending K-Epsilon	29,7 Pa
K-Omega SST	29,3

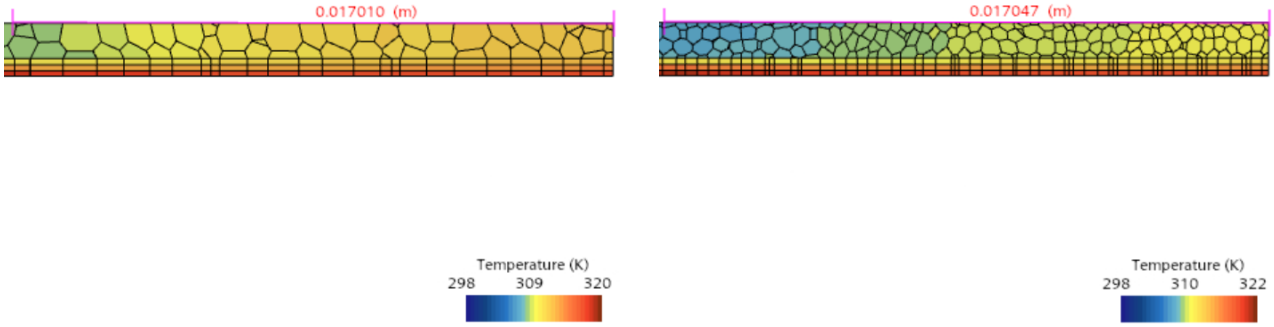
At first impression it seems that the best case is the Spalart-Allmaras turbulence model. But at the moment of the turbulence model validation the y+ value was also extremely low for all models which caused a high and also unwanted computational demand. Once the y+ treatment and the mesh study began there was a phenomena where the pressure drops of Spalart-Allmaras and of the Realizable two layer k-Epsilon did not converge while the Elliptic Blending k-Epsilon turbulence model converged consistently. Therefore the Elliptic Blending k-Epsilon turbulence model is chosen for the ongoing simulations.

Once the turbulence model is selected a mesh study has been conducted in order to see which of the following meshes suits best for this case. In table 4 we can find a detailed configuration of the mesh with the relevant values that has to be compared and the total cell number.

Table 4: Mesh configurations

	Meshtype	**Base size**	**T_average**	**Delta_T**	**Delta_P**	**Total cell number**
Fluid	Polyhedral Cell	5 mm				
Plate	Trimmed Cell	1 mm	35.53 K	8.82 K	28.6 Pa	509,049
LIB	Trimmed Cell	10 mm				
Fluid	Polyhedral Cell	10 mm				
Plate	Polyhedral Cell	8 mm	73.03 K	51.59 K	28.4 Pa	317,907
LIB	Polyhedral Cell	20 mm				
Fluid	Polyhedral Cell	5 mm				
Plate	Polyhedral Cell	8 mm	72.49 K	51.9 K	28.45 Pa	459,452
LIB	Polyhedral Cell	20 mm				
Fluid	Polyhedral Cell	8 mm				
Plate	Polyhedral Cell	8 mm	72.91 K	52.06 K	28.25 Pa	331,640
LIB	Polyhedral Cell	20 mm				
Fluid	Polyhedral Cell	15 mm				
Plate	Polyhedral Cell	8 mm	73.13 K	51.78 K	28.98 Pa	256,259
LIB	Polyhedral Cell	20 mm				
Fluid	Polyhedral Cell	0.4 mm				
Plate	Polyhedral Cell	8 mm	74.01 K	57.29 K	28.61 Pa	887,152
LIB	Polyhedral Cell	2 mm				

The geometry is implemented with the polyhedral mesh for both, the solids and the fluid parts. The polyhedral mesh has a major advantage compared to other meshes, since it has a large number of neighbors. In this way, it can approximate gradients and predict the flow distribution better [8]. Even though the last configuration has the biggest cell number of almost 900 thousand cells and increases the computational time this configuration is the nearest one to the values of the literature (see Table 1). This can be explained by the fact that the polyhedral mesh is used and also because if by looking at Figure 6 the difference between a coarse mesh and a fine mesh can be seen. This figure shows the temperature from the cross section of the fluid with the length of 17 mm. Figure 6a shows the cross section of the third mesh configuration from Table 4 while Figure 6b shows the last mesh configuration also from Table 4. If both figures are compared it can be seen that the fine mesh is calculating the temperature behaviors faster. This is the fact that the the solver has to calculate over smaller distances getting faster and better solutions. The coarse mesh needs more cells in order to get similar results because the distance from one cell to another is greater and the solver can not calculate it as fast as the smaller step sizes.



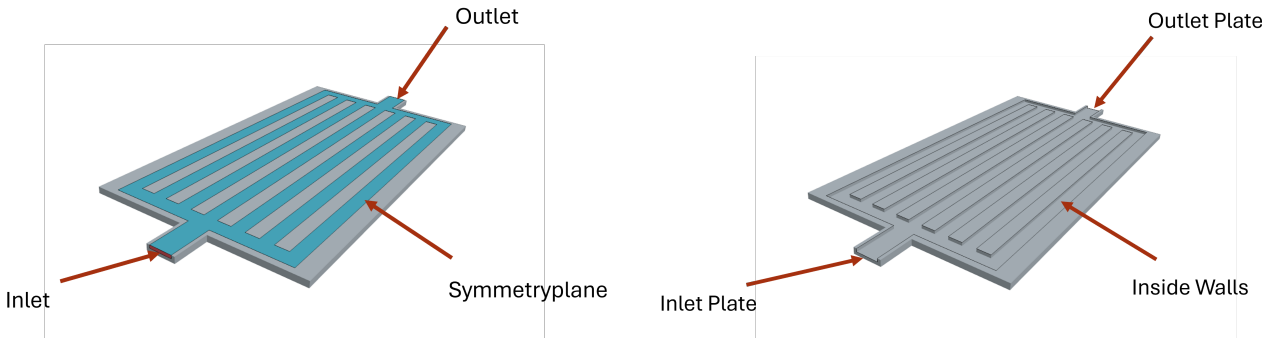
(a) Coarse cross section mesh.

(b) Fine cross section mesh.

Figure 6: Comparison of cross section at the fluid domain.

As mentioned before we also need to select specific faces for the simulations. This is for three different reasons. The first is to declare the correct regions, for example for the inlet and outlet. The second reason is to get mesh refinement on specific points of the geometries in order to get a conformal mesh at the contact faces and also for better calculations at specific points. The last reason is to select the same faces in every iteration of the automated java script so that it can redefine the first two important reasons for different geometries. Figure 7 shows which faces are selected in order to get these improvements.

A conformal mesh is when two faces perfectly align with each other creating the mesh nodes on the same point. This will decrease the computational time since it will not need to use interpolations or averaging and will also reduce numerical errors.[9]



(a) Faces from the Fluid.

(b) Faces from the plate.

Figure 7: Face selection for improved simulation.

After an accurate change of setting parameters on the prism Layer control of the fluid mesh, the final parameters stated on Table 5 are defined. In Figure 8 we see that y+ is mostly over 0.5. With this it can be state, that we are not too small on y+ which will cause an increase

of computational time but also that we are not too big, since we are using Elliptic Blending K-Epsilon turbulence model and want to be close by 1 for a good wall treatment [10].

Table 5: Prism Layer Controls

Properties	**Parameter**
Number of Prism Layers	3
Prism Layer Stretching	1.1
Prism Layer Reduction	10.0 %
Concave Angle Limit	90.0 deg
Convex Angle Limit	360.0 deg
Near Core Aspect Ratio	0.8
Gap Fill	25.0 %
Minimum Thickness	10.0 %
Boundary March Angle	50.0 deg
Prism Layer Total Thickness	0.5 mm (absolute size)

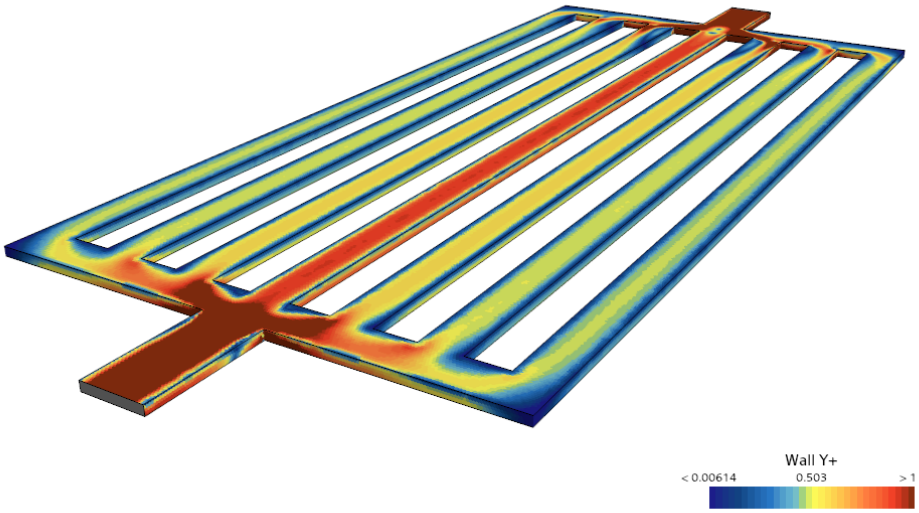


Figure 8: y+ distribution across the surface.

With all these steps, the validated geometry is very close to the study of Wu, [1]. In Table 6 a comparison of the values from the literature (center) with our values (right) is presented. The temperature has a difference about 1°C while the Delta P has an offset of about 3.5 Pa.

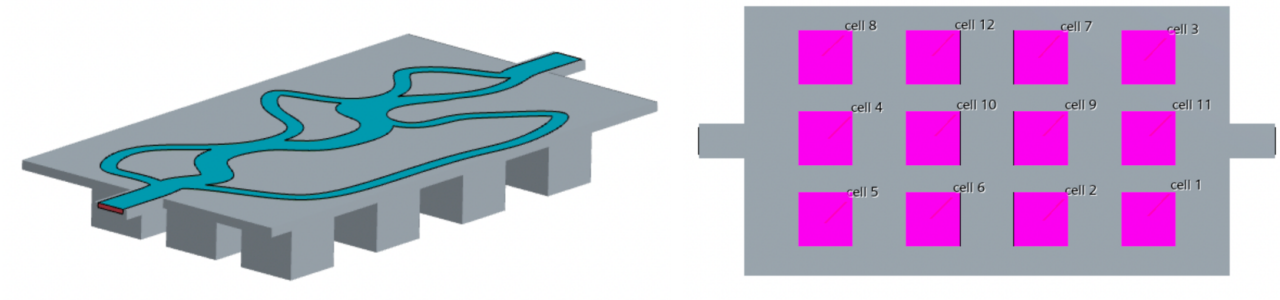
Table 6: Comparison Wu's study with our results

Values	**Literature Values**	**Simulated Values**
T_average	74.51 °C	75.81 °C
Delta_T	56.86 °C	57.79 °C
Delta_P	32.02 Pa	28.44 Pa

3.2.2 Heat Configuration

The simulations ran for four different heat configurations. In each configuration, heat is supplied from all 12 cells, with varying volumetric heat rates between each configuration. These

configurations are intended to represent a non-uniform heat influx. Figure 9 visualizes the complete system which is being simulated.



(a) Isometric view. (b) Bottom view.

Figure 9: Different views of the system.

Each cell is cubic with a volume of $8{\times}10^3$ mm^3. The 12 cells form a 3×4 matrix on the back of the plate. Table 7 shows the volumetric heat produced by the different cells.

Table 7: Topology of the heat distribution of the cells.

Configuration	**2.5** $\times 10^6$ **W/m**^{3}	**3.75** $\times 10^6$ **W/m**^{3}	**5** $\times 10^6$ **W/m**^{3}
Layout 1	{1, 2, 3, 4, 5, 6}	{7, 8, 9, 10}	{11, 12}
Layout 2	{7, 8, 9, 10, 11, 12}	{1, 2, 3, 4}	{5, 6}
Layout 3	{1, 3, 5, 7, 9, 11}	{6, 8, 10, 12}	{2, 4}
Layout 4	{2, 4, 6, 8, 10, 12}	{3, 5, 7, 9}	{1, 11}

3.2.3 Data Export

After each simulation, the temperature and heat distribution are saved as CSV files using a representation grid over the plate. This allows the temperature and heat values to be determined at each point in the grid, providing a 2D representation of the distribution. The geometry is also exported as a CSV file, using the representation grid and a user defined function that represents the geometry based on density. If the density at a specific point in the grid is higher than 2000 kg/m^3, that point is assigned a value of 1; otherwise, it is assigned 0. The points with lower density, representing the coolant, are assigned 0, while the points with higher density corresponds to the aluminium plate. In this way the geometry is represented in 2D within the CSV file.

A Java macro was created to automate the simulations, allowing all geometries to be simulated using four heat configurations and four different inlet velocities. When the simulation had reached maximum steps of 700, both the CSV data and the simulation files were saved.

3.3 Surrogate Model

The surrogate model predicts pressure drop and temperature distribution given the cooling channel geometry, inlet velocity and heat source configuration. By using a neural network trained on CFD simulation data, the surrogate model provides rapid estimations, significantly reducing the computational cost and time required for evaluating different designs. This enables efficient exploration of the design space during the optimization process.

3.3.1 Data Preprocessing

The data was read from CSV files and split into training (80%, corresponding to 3200 samples) and testing (20%, corresponding to 800 samples) datasets, ensuring no overlapping geometries. The preprocessing steps included normalizing the inlet velocity, temperature, pressure drop, and heat source to a range between 0 and 1. The channel geometry did not require normalization, as the data is already binary (1 represents fluid, and 0 represents solid).

Additionally, outliers were removed based on pressure drop values to ensure a uniform distribution. A small number of samples exhibited significantly higher pressure drop values, which could skew the normalization process and suppress the majority of other samples. These unrealistically high pressure drop values were caused by unrealistic channel geometries. To address this, the top 5% of samples with the highest pressure drop values were completely removed from the dataset. This removal included all associated variables-namely, inlet velocity, geometry, and temperature-for the affected samples, resulting in a more uniform distribution of pressure drop values.

Finally, the datasets were augmented by creating two samples from each original sample through mirroring the geometry, heat source, and temperature matrices along the x-axis while keeping the pressure drop and inlet velocity unchanged. This augmentation process doubled the size of the training dataset.

3.3.2 Model Architecture

For the surrogate model, we developed a custom neural network architecture shown in Figure 10. Since the problem involves learning mappings between inputs and outputs governed by PDEs, we used FNOs in this network. The network takes three input channels: a geometry channel (binary matrix), a heat source matrix, and an inlet velocity scalar expanded to a scalar matrix to match the size of the other channels. The network outputs a 2D temperature distribution matrix and a scalar pressure drop value.

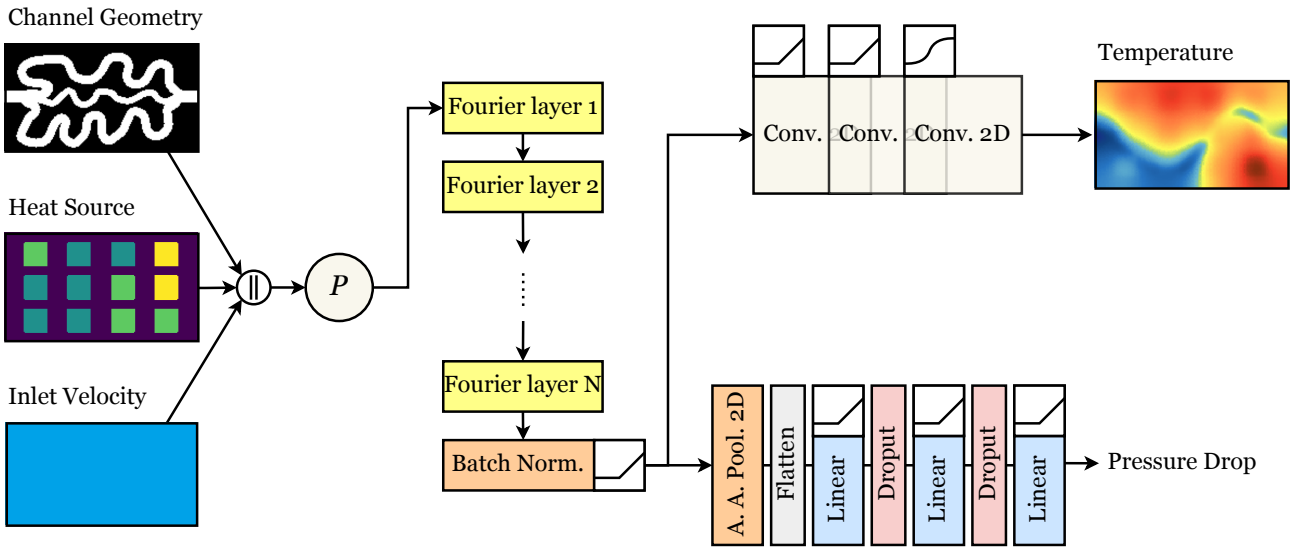


Figure 10: Neural network architecture of the SM, which takes input channels for geometry, heat source, and inlet velocity, and outputs both the temperature distribution and pressure drop. The small white squares represent different activation functions.

The three input channels are concatenated (||) and passed through a convolutional layer to increase the feature dimensions, which essentially is a projection (P) into a higher dimension.

This is followed by Fourier layers, which help the network capture complex spatial patterns by transforming the data into the frequency domain.

After the Fourier layers, batch normalization is applied to speed up training and stabilize the learning process. The network then splits into two branches: one for predicting pressure drop and the other for predicting temperature distribution.

Pressure Drop Branch: This branch uses adaptive average pooling to reduce the spatial dimensions while keeping important information. The pooled output is flattened and passed through three fully connected layers with ReLU activation and dropout between them to prevent overfitting. The layers progressively reduce the number of neurons, ending with a single scalar output for the pressure drop. A final ReLU activation ensures the pressure drop value is positive.

Temperature Branch: This branch consists of three convolutional layers. The first two layers use ReLU activations to capture non-linearity, while the last layer uses a Sigmoid activation to normalize the temperature values between 0 and 1.

3.3.3 Loss Function

The surrogate model is trained by minimizing a combined loss function $\mathcal{L}$ that accounts for both the scalar pressure drop and the 2D temperature distribution predictions. Since all data are normalized, we define the loss function to be the weighted sum of the Mean Squared Errors (MSE) for each output:

$$\mathcal{L} = w_p \text{MSE}(p_{\text{pred}}, p_{\text{true}}) + w_t \text{MSE}(T_{\text{pred}}, T_{\text{true}}) \tag{1}$$

The weights were chosen as $w_p = w_t = 0.5$, ensuring that the total loss value is theoretically bounded within $[0, 1]$. The MSE for pressure drop is calculated as:

$$\text{MSE}(p_{\text{pred}}, p_{\text{true}}) = \frac{1}{N} \sum_{j=1}^{N} \left(p_{\text{pred},j} - p_{\text{true},j}\right)^2, \tag{2}$$

and the MSE for the temperature distribution is computed pixel-wise as:

$$\text{MSE}(T_{\text{pred}}, T_{\text{true}}) = \frac{1}{NP} \sum_{j=1}^{N} \sum_{i=1}^{P} \left(T_{\text{pred},j,i} - T_{\text{true},j,i}\right)^2. \tag{3}$$

Here, N denotes the number of samples, and P denotes the number of pixels in each 2D temperature matrix. $p_{\text{pred},j}$ and $p_{\text{true},j}$ are the predicted and true normalized pressure drop values for the j-th sample, respectively. $T_{\text{pred},j,i}$ and $T_{\text{true},j,i}$ are the predicted and true normalized temperature values at the i-th pixel of the j-th sample. Minimizing this loss function will reduce the prediction errors for both pressure drop and temperature distribution.

3.3.4 Training & Hyperparameter Tuning

The goal of hyperparameter tuning is to improve the performance of a machine learning algorithm. There are several approaches to find an optimal configuration of hyperparameters. In our case, we computed all possible configurations of hyperparameters and selected 300 random combinations which then were trained and compared. Due to increased demand of computational resources, some of the 300 selected couldn't be trained.

Table 8: Summary of tuned hyperparameters and their search space.

Hyperparameter	**Search Space**
Number of modes for the FNO layers.	{20, 40, 60}
Number of hidden channels in each FNO layer.	{64, 256, 512}
Number of Fourier Neural Operator blocks.	{1, 2, 3}
Number of neurons in the first fully connected layer.	{512, 1024, 2048}
Number of neurons in the second fully connected layer.	{64, 128, 512}
Activation function for intermediate layers.	{ReLU, LeakyReLU, Softsign}
Dropout rate for regularization.	{0.2, 0.4, 0.6}
Activation function for the output layer.	{GELU, ReLU, Softsign}

3.4 Geometry Optimization

This section introduces the methodology used for optimizing geometric designs in a multi-objective context. The following subsections detail the implementation, including encoding strategies, mutation mechanisms, and evaluation processes.

3.4.1 Evolutionary Algorithm

The implementation of EA in this paper primarily follows the NSGA-II algorithm, including the steps: Evaluation, Selection, and Mutation. This approach excludes crossover to address specific balancing issues related to exploration and exploitation within the algorithm. This implementation has been chosen over classical EA methods because it has been shown to yield superior results for multi-objective fitness functions [7]. To implement this algorithm, the evaluation and the mutation along with the encoding of an individual, must be designed to ensure convergence to an optimal solution.

The first step is to define the encoding of an individual. For this, the design used for data generation is reused. Each geometry is designed using random points on the plate, which are then connected through a spline function. Additionally, the geometry must be flexible enough to accommodate multiple channels. Therefore, each individual is encoded into a matrix of size $[n_{\text{channels}}, n_{\text{points}}]$. Initially, each individual in the population contains only a single channel, which can subsequently increase through mutation.

The design of the mutation mechanisms is vital for this algorithm, as crossover has been removed. Both mutation and crossover serve to balance exploration and exploitation throughout the optimization run. Without properly balancing these two phenomena, there is a risk of losing or never finding the most optimal solution. As mentioned earlier, the mutation mechanism is also the only way for the geometry to generate additional channels. The design implemented for this project contains three different mechanisms: adding noise to the random points, splitting a channel at a random point, and merging two channels at a random point. Each of these mechanisms is assigned a specific probability of occurring.

During the evaluation phase, the surrogate model is implemented. Each individual in the population is evaluated by translating the encoding into a grid, which is then used as input. The maximum temperature is extracted from the predicted heat distribution and, along with the predicted pressure drop, serves as the two objectives. In the NSGA-II, the population is then divided into sets of solutions based on their dominance levels [7]. Additionally, NSGA-II calculates the crowding distance to determine which individuals are selected for the next generation, thereby expanding the search area. The idea here is to prioritize solution with a higher crowding distance so as to keep the more unique solutions and avoid clustering.

4 Results

In the following chapter, we will discuss the results of the surrogate model, as well as the hyperparameter tuning and finally the results of the geometry optimization can be found.

4.1 Surrogate Model

The analysis searching for the best combination resulted in the combination shown in Table 9:

Table 9: Summary of optimal hyperparameters.

Hyperparameter	**Optimal Parameter**
Number of modes for the FNO layers.	{20}
Number of hidden channels in each FNO layer.	{64}
Number of Fourier Neural Operator blocks.	{3}
Number of neurons in the first fully connected layer.	{1024}
Number of neurons in the second fully connected layer.	{64}
Activation function for intermediate layers.	{LeakyReLU}
Dropout rate for regularization.	{0.6}
Activation function for the output layer.	{GELU}

By comparing the worst combination of tested hyperparameters with the best performing combination, you can see the relevance of tuning the hyperparameters in Figure 11.

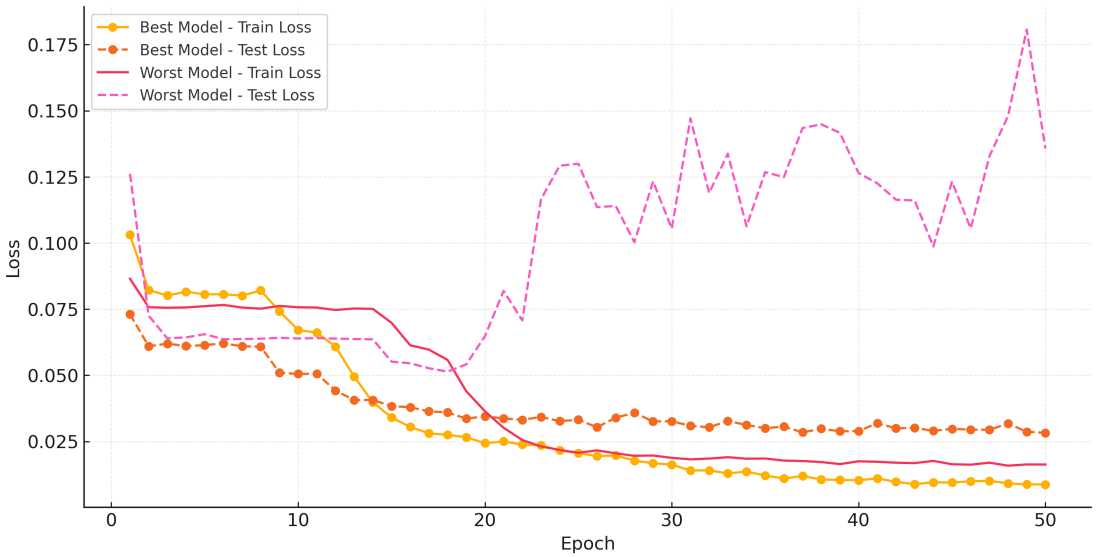


Figure 11: Comparison of train and test loss over epochs for best and worst hyperparameter configurations.

4.1.1 Performance

We performed a comprehensive evaluation on 300 test samples to gain a broader perspective of the model's performance. For each sample, two key metrics were calculated: the MAE between the predicted and ground-truth temperature fields, and the absolute difference between the predicted and ground-truth pressure drops. Across all 300 samples, the temperature MAE had a median of 2.53 K with a standard deviation of 3.74 K, while the pressure drop difference showed a median of 10.5 Pa and a standard deviation of 37.4 Pa. Figure 12 illustrates the distributions of both metrics, with histograms overlaid by kernel density estimate curves.

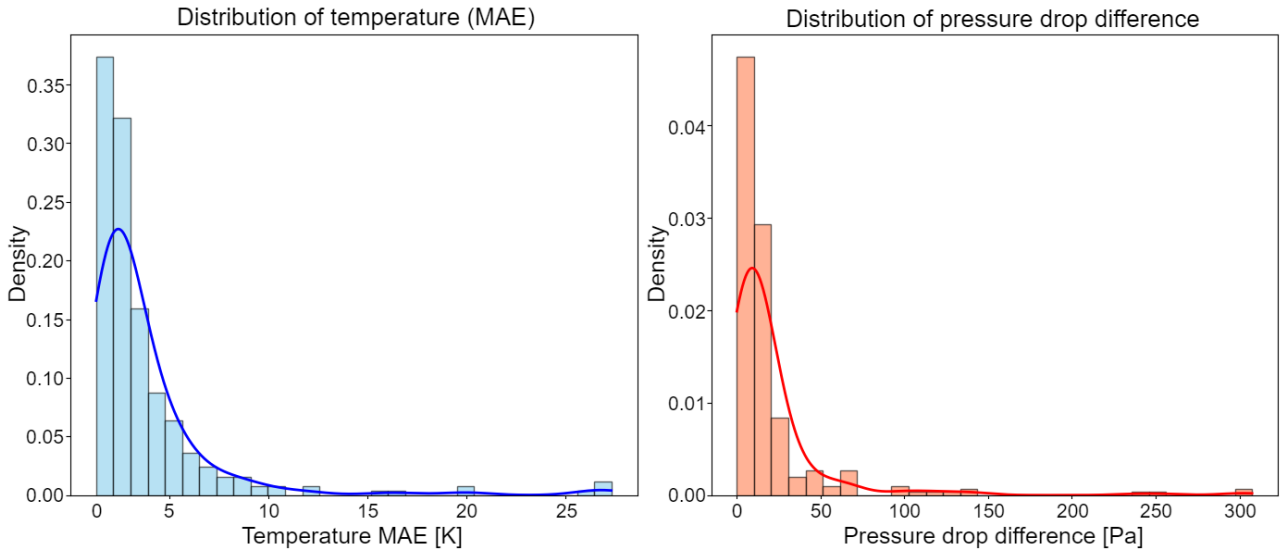


Figure 12: Distributions of the temperature MAE (left) and the pressure drop difference (right) for the 300 test samples. Each histogram is overlaid with its kernel density estimate curve.

Both the temperature MAE and the pressure drop difference appears to cluster around its median, however there is a noticeable tail extending toward higher values. Overall, these findings indicate that while the model achieves satisfactory predictions on most test samples, it performs significantly worse on certain atypical cases.

4.1.2 Sample Prediction

In Figure 13, three arbitrarily selected samples from the test dataset are shown. For each sample, we display the ground-truth temperature, the predicted temperature and the corresponding error map. The predicted and true pressure drops for each sample are as follows:

- Top sample: Predicted = 450 Pa, True = 438 Pa
- Middle sample: Predicted = 197 Pa, True = 201 Pa
- Bottom sample: Predicted = 111 Pa, True = 94 Pa

The mean absolute errors for the temperature distributions (averaged over the entire map) are 2.45 K, 2.08 K, and 2.23 K for the top, middle and bottom samples, respectively.

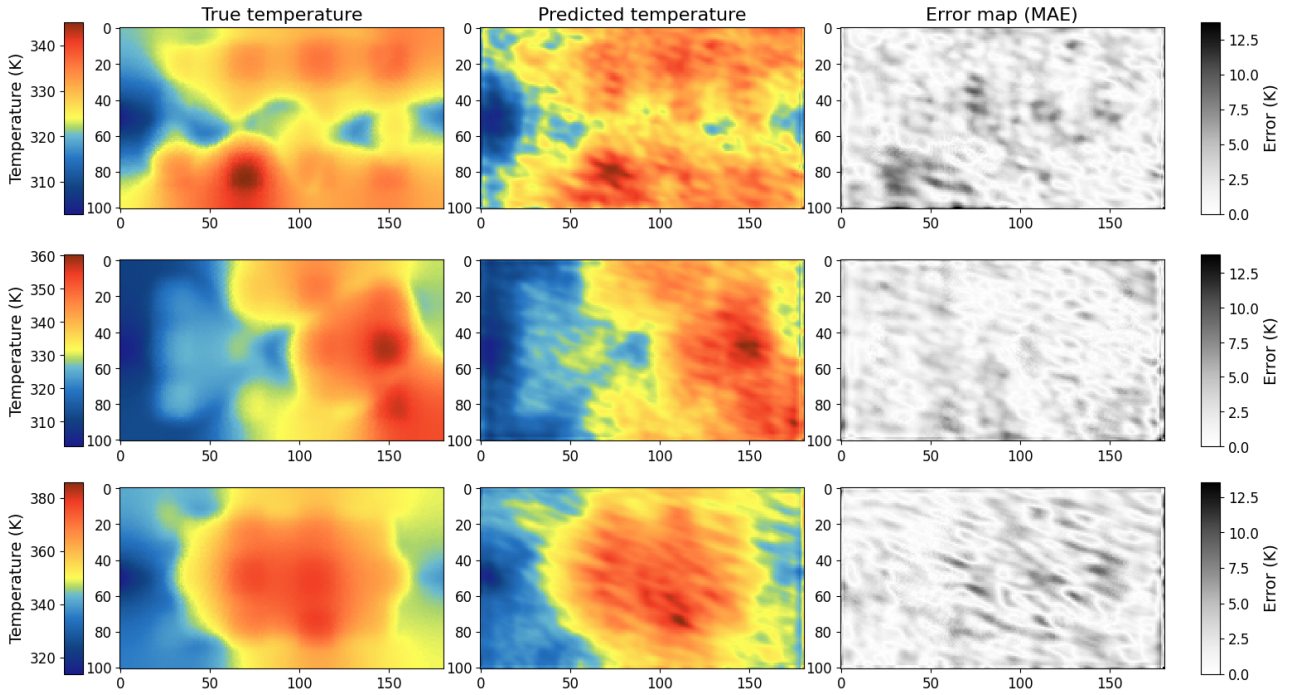


Figure 13: Examples of ground-truth and predicted 2D temperature distributions, along with their MAE maps, for three test samples.

4.2 Geometry Optimization

As described above, the output from the EA is a Pareto front, see Figure 14. The Pareto front demonstrates the set of non-dominated solutions achievable by the algorithm, showcasing the balance between minimizing pressure drop and maximum temperature. From this set of solutions, the optimal solution for each objective can be extracted, along with a balanced solution that considers both objectives. The geometries corresponding to all three solutions are presented below in Figures 15 and 16. The geometries have been optimized using an inlet velocity of $v = 0.1\,\mathrm{m/s}$ and the second heat configuration described in Section 3.2.2.

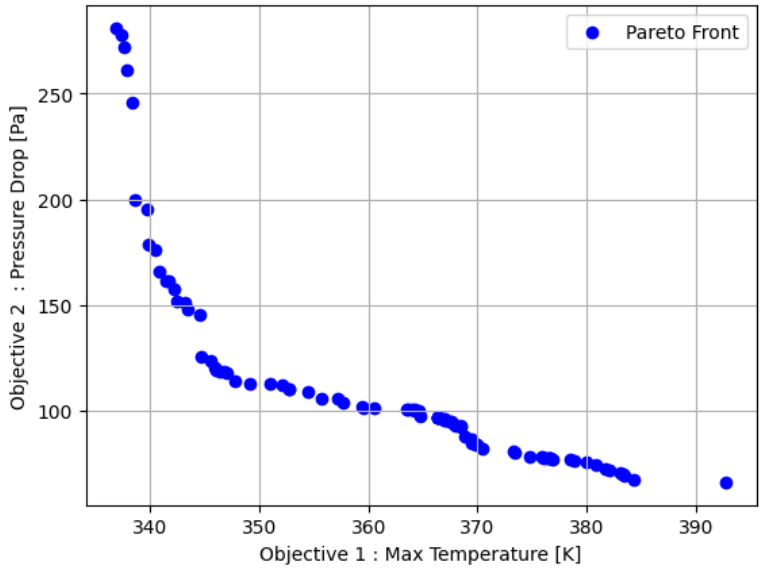


Figure 14: Pareto Front generated by the EA. The graph illustrates the spread of solutions from the Pareto Front in the objective space with Maximum Temperature on the x-axis and Pressure Drop on the y-axis. Each point represents an individual solution.

Figure 14 shows us the range of objective values that are obtainable by the EA. The lowest maximum temperature over the plate is approximately 336.92 K while the lowest pressure drop is 66 Pa. See Figure 15 to for the geometries with these values.

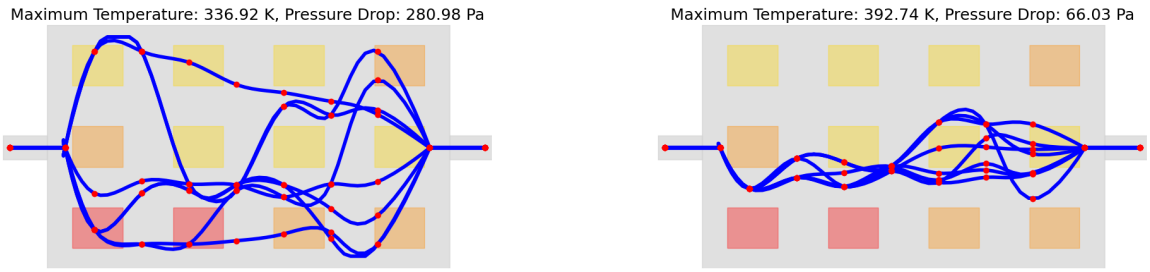


Figure 15: Comparison of Solutions from the Pareto Front. The left shows the geometry that minimizes the maximum temperature, while the right shows the geometry that minimizes the pressure drop.

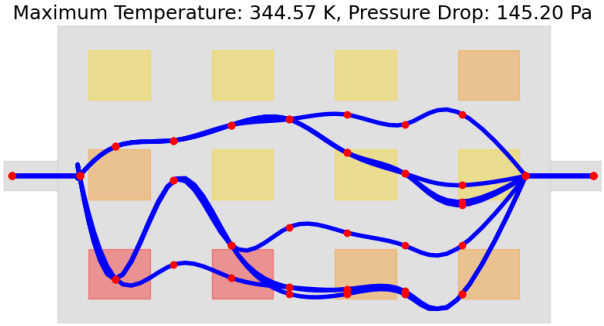


Figure 16: A balanced solution from the Pareto front. The image shows the geometry from the Pareto Front with a balance towards both objectives.

5 Discussion

This section explores the performance of the surrogate model and the geometry optimization process. We analyze how these components contribute to the optimization of cooling channel designs and assess their respective strengths and limitations.

5.1 Surrogate Model

The results for the surrogate model show that this framework indeed fulfills all the requirements initially proposed. With a median MAE of 2.53 K for temperature and a median pressure drop difference of 10.5 Pa across the test samples, the model reliably estimates key performance metrics, enabling rapid evaluation of numerous design variations.

Integrating the surrogate model into the optimization framework enhances the efficiency of the design process, allowing for extensive exploration of the design space of optimal cooling solutions that balance thermal regulation and pressure drop. While the model performs well for most cases, certain outliers indicate areas where predictions deviate from CFD results. Nevertheless, the surrogate model proves to be a valuable tool in accelerating the design and optimization of EV battery cooling systems, contributing to more innovative and effective thermal management strategies.

5.2 Evolutionary Algorithm

The results of the geometry optimization demonstrate the promising effectiveness of NSGA-II in addressing the design problem. While the solutions are not yet perfect, NSGA-II is capable of finding meaningful solutions that balance both objectives. Figure 14 illustrates the trade-off relationship between the two objectives. Notably, the Pareto front is concave rather than linear. This concave shape indicates that the algorithm effectively finds solutions that optimize one objective without proportionally impacting the other. In contrast, a linear Pareto front does not allow for such balanced optimization, resulting in the balanced solution losing meaningful value. Additionally, Figure 14 displays the range of objective values as mentioned earlier. These ranges are derived from the extreme cases. The results indicate that the optimal solution for temperature is approximately 336 K, and the corresponding pressure drop is around 66 Pa. Both values are comparable to those reported by Wu [1] showing the results are indeed in the desired range.

The first extreme case, representing the solution with the lowest maximum temperature, demonstrates well-distributed channels that cover every cell, as shown in Figure 15. To cover every cell, numerous individual channels are split with significant distances between them, resulting in a very high pressure drop. The relatively extreme pressure drop makes this solution less practical for applications.

The second extreme case displays the lowest pressure drop in the Pareto front. As shown in Figure 15, this is achieved by concentrating all channels in the center, effectively creating a single thick channel in the CAD geometry. This concentration leads to inefficiencies in thermal management. The maximum temperature for this geometry is approximately 392 K, which is well above an efficient range. Similarly to the first extreme case, this solution is not practical.

An example of a balanced solution is shown in Figure 16. This is just one example among many with similar performance, as observed from the Pareto front, see figure 14. This solution achieves thermal efficiency similar to that in the first extreme case by spreading out the channels while significantly improving the pressure drop. This improvement is achieved by minimizing the total number of distinct channels at any point across the plate. Consequently, this solution has at most four channels, compared to the six distinct channels shown in Figure 15.

6 Potential Future Improvements

While the explored framework demonstrates advancements in optimizing cooling channel designs, there are several steps for further enhancement. This section explores potential improvements in geometry generation, surrogate modeling and the evolutionary algorithm.

6.1 Geometry Generation

One possible improvement for the geometry generation is to apply different path generation algorithms. The random points and spline algorithm used in this project is very simple but effective. Another algorithm that might be interesting to implement is the random-walk algorithm, in which an agent walks on a grid on the cooling plate from the inlet to the outlet, thus creating a random path.

Additional adjustable parameters can be added to the algorithm. For instance varying channel profile. In the project the channel width is a set value based on the rectangular profile used in the sweep function in FreeCAD. By implementing varying path profiles the randomness and robustness of the dataset is increased which in turn could improve the surrogate model.

The boundary margin parameters for parametric modeling can be further refined to enhance performance. Currently, as described in Chapter 3.1.1, the margin between the channels and the

plate boundaries is defined as the distance between random points and the plate edges. However, the spline curves connecting these random points are not constrained, leading to unpredictable behavior. If the margin is too small, the spline curves may cross the plate boundaries, causing design violations. Conversely, excessively large margins may result in the channels covering less of the cooling plate, thereby reducing cooling efficiency. Introducing a margin constraint that directly governs the behavior of the spline curves could address this issue and improve the overall design robustness.

6.2 Surrogate Model

The size of the dataset can always be increased by increasing the number of CFD simulations that are used for training the surrogate model. Increasing the dataset with different types of channel geometries allows the surrogate model to become more general, which will improve the prediction performance for unseen geometries.

Additionally, incorporating more diverse operating conditions, such as varying inlet velocities and heat source distributions, can enhance the model's robustness. Exploring advanced neural network architectures or ensemble methods may further improve performance and reliability. Lastly, implementing techniques like cross-validation and regularization can also help in mitigating overfitting, ensuring the model maintains high performance across a broader range of scenarios.

6.3 Evolutionary Algorithm

The EA has shown good results but has much room for improvements and expansion. The performance of this algorithm is always connected to surrogate model, and as such all improvements made to all other parts of this framework would directly lead to more reliable results for this algorithm. The algorithm for now has been restricted for the purpose of getting initial results however EA often benefit from freedom in the decision making and thereby expanding the search space within the objective space. For example the thickness of each channel is kept constant but could very much be a decision parameter for the algorithm to decide on. Similarly many other parameters can be added for the system to optimize. Other improvements could be to add more objective for the algorithm to balance, for example the average temperature across the plate T_{avg}.

References

[1] Mao-Sung Wu. "Multi-objective topology optimization of cold plates featuring branched and streamlined mini-channels for thermal management system of lithium-ion battery module". In: *Journal of Energy Storage* 72 (2023). ISSN: 2352-152X. DOI: 10.1016/j.est.2023.108362. ScienceDirect: S2352152X23017590. URL: https://doi.org/10.1016/j.est.2023.108362.

[2] Xianqi Chen et al. "A Deep Neural Network Surrogate Modeling Benchmark for Temperature Field Prediction of Heat Source Layout". In: *CoRR* abs/2103.11177 (2021). arXiv: 2103.11177. URL: https://arxiv.org/abs/2103.11177.

[3] Yanfang Lyu et al. "Multi-fidelity prediction of fluid flow and temperature field based on transfer learning using Fourier Neural Operator". In: *Preprint* (2023). arXiv:2304.06972. URL: https://arxiv.org/abs/2304.06972.

[4] Wang Xiao et al. "Fourier neural operator based fluid-structure interaction for predicting the vesicle dynamics". In: *Preprint* (2024). arXiv:2401.02311. URL: https://arxiv.org/abs/2401.02311.

[5] Zongyi Li et al. "Fourier Neural Operator for Parametric Partial Differential Equations". In: *arXiv preprint arXiv:2010.08895* (2020).

[6] Nikola Kovachki et al. "Neural Operator: Learning Maps Between Function Spaces". In: *arXiv preprint arXiv:2104.02540* (2021).

[7] Kalyanmoy Deb et al. "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. DOI: 10.1109/4235.996017.

[8] Siemens Digital Industries Software Community. *Polyhedral Meshing in STAR-CCM+*. Accessed: 2024-10-26. 2020. URL: https://community.sw.siemens.com/s/article/JC-4-332.

[9] Christoffer Johansson, M.Sc. *Conformal meshing in Simcenter STAR-CCM+*. Accessed: 2025-01-09. 2023. URL: https://volupe.com/simcenter-star-ccm/conformal-meshing-in-simcenter-star-ccm/.

[10] Siemens Digital Industries Software Community. *Which y+ should I choose for which turbulence model?* Accessed: 2024-10-01. 2023. URL: https://community.sw.siemens.com/s/article/EK-5-628.

A Relevant Code

All code used for this study, including data preprocessing, model training, and evaluation is available at `https://github.com/FrancoFusion/ML-CFD`.