POLITECNICO
MILANO 1863

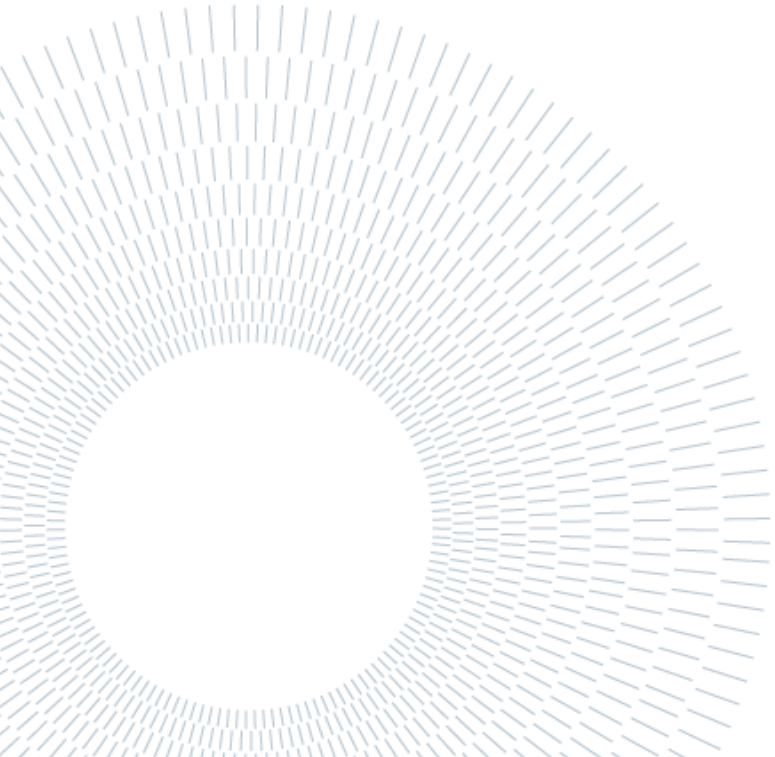
SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Critical Heat Flux Prediction by Ensemble and Physics-Informed Neural Networks

TESI DI LAUREA MAGISTRALE IN
NUCLEAR ENGINEERING
INGEGNERIA NUCLEARE

Author: **Irene Gatti**

Student ID: 10590389
Advisor: Prof. Enrico Zio
Co-advisor: Prof. Ibrahim Ahmed
Academic Year: 2023-2024



Abstract

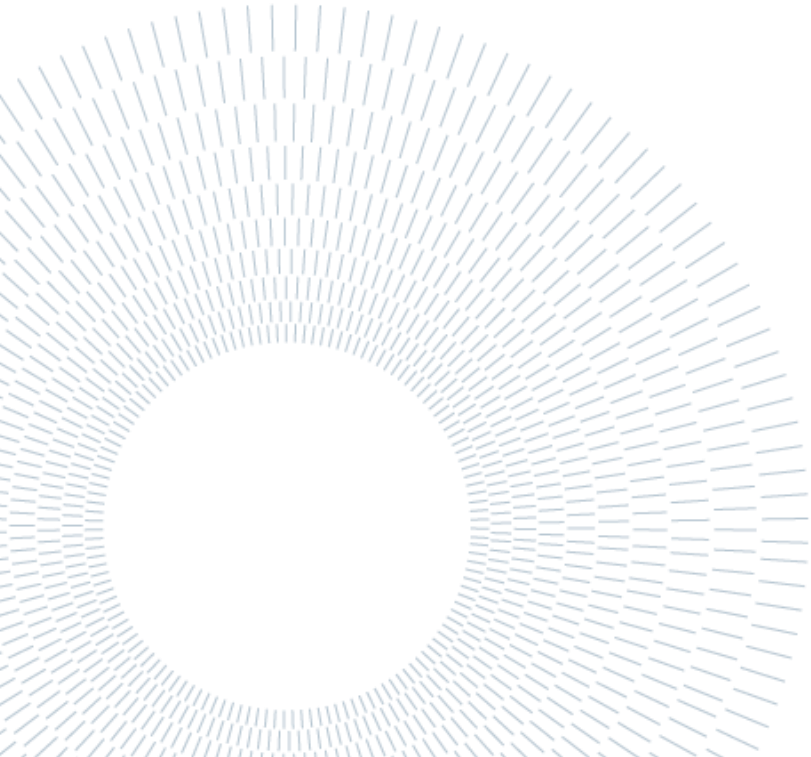
Critical Heat Flux (CHF) relates to the transition from nucleate boiling to film boiling. This transition leads to a significant decrease in heat transfer efficiency. Consequently, accurate CHF prediction is essential for the safety and performance of water-cooled nuclear reactors where thermohydraulic margins are critical. Over the years, numerous CHF prediction models have been developed, including mechanistic, empirical, artificial intelligence (AI) and machine learning (ML). This work proposes (1) a novel ensemble of neural networks (NNs) model aimed at enhancing accuracy by combining individually optimized NNs with varied architectures and hyperparameters and (2) an interpretable physics-informed neural network (PINN) that integrates governing physical laws into the learning process to improve prediction accuracy. For the ensemble model, systematic procedures are presented to identify and optimize the best NNs and to aggregate them into the optimal ensemble. As for the PINN, the Westinghouse (W-3) correlation, an empirical CHF correlation for water-cooled reactors, is integrated as a physical model to drive the learning process. Such correlation is simple to implement and effective for both subcooled and saturated boiling conditions. To interpret the PINN results, SHapley Additive exPlanations (SHAP) and Local Interpretable Model-Agnostic Explanations (LIME) are used, providing insights into the influence of input variables compared to standard NN models. The two prediction models developed are validated using experimental CHF data made available by the Working Party on Scientific Issues and Uncertainty Analysis of Reactor Systems (WPRS) Expert Group on Reactor Systems Multi-Physics (EGMUP) task force on AI and ML for Scientific Computing in Nuclear Engineering projects, promoted by the OECD/NEA. The obtained results demonstrate that the proposed models outperform state-of-the-art models, with sensitivity analysis further confirming robustness across various input parameters.

Key-words: critical heat flux, nuclear reactors, neural network, physics-informed neural network, ensemble model, LIME and SHAP

Abstract in italiano

Il Critical Heat Flux (CHF) si riferisce alla transizione tra nucleate boiling e film boiling. Questa transizione porta a una significativa riduzione dell'efficienza di trasferimento del calore. Di conseguenza, una previsione accurata del CHF è essenziale per la sicurezza e le prestazioni di reattori nucleari raffreddati ad acqua, dove i limiti termo-idraulici sono fondamentali. Nel corso degli anni sono stati sviluppati numerosi metodi per predire il CHF, tra cui modelli meccanicistici, empirici, di intelligenza artificiale (IA) e di machine learning (ML). Questo lavoro propone (1) un nuovo modello d'insieme di reti neurali (NNs) progettato per migliorare le prestazioni combinando modelli individualmente ottimizzati con architetture e iperparametri diversi, e (2) un Physics-Informed Neural Network (PINN), che integra la fisica del fenomeno nel processo di apprendimento della rete per migliorarne l'accuratezza. Per il modello d'insieme, vengono presentate procedure sistematiche per identificare i migliori modelli di NN e per aggregarli una volta ottimizzati in un ensemble ottimale. Nell'approccio PINN, la correlazione Westinghouse (W-3), una correlazione empirica del CHF per reattori raffreddati ad acqua, viene integrata come modello fisico nel processo di apprendimento. Questa correlazione è semplice da implementare ed è efficace sia per condizioni di ebollizione sottoraffreddata che satura. Per interpretare i risultati del PINN, vengono utilizzati i metodi SHAP (SHapley Additive exPlanations) e LIME (Local Interpretable Model-Agnostic Explanations), che forniscono informazioni sull'influenza delle variabili di input rispetto ai modelli standard di NN. I due modelli di previsione sono stati validati utilizzando dati sperimentali sul CHF messi a disposizione dal Working Party on Scientific Issues and Uncertainty Analysis of Reactor Systems (WPRS) Expert Group on Reactor Systems Multi-Physics (EGMUP), che ha promosso progetti di IA e ML per il calcolo scientifico in progetti di ingegneria nucleare, promosso dall'OECD/NEA. I risultati ottenuti dimostrano che i modelli proposti superano gli attuali metodi all'avanguardia, e un'analisi di sensitività conferma ulteriormente la robustezza di questi modelli rispetto a vari parametri input.

Parole chiave: critical heat flux, reattori nucleari, neural network, physics-informed neural network, modello d'insieme, LIME e SHAP



Contents

List of Figures

List of Tables

List of Symbols

Variable	Description	SI unit
D	tube diameter	m
L	heated length	m
P	pressure	kPa
G	mass flux	kg/s/m^2
X	outlet quality	-
T	temperature	°C
H_{sat}	saturation enthalpy	kJ/kg
H_{in}	inlet enthalpy	kJ/kg
H_{sub}	inlet subcooling	kJ/kg
q''	surface heat flux	kW/m^2
y	Input vector	
W	weight matrix	
b	bias vector	
f	activation function	
i	number of rows index	
N	total number of rows	
$\hat{q}$	predicted output	
q	Actual output	
$\mathcal{L}$	loss function	
α	learning rate	
θ	model parameters	
λ	tuning hyperparameter	
r	residual	
$\hat{\mu}$	mean value	
Q^2	coeff. of determination	

Acronyms	Description
DNB	departure from nucleate boiling
DO	dryout
CHF	critical heat flux

NRC	nuclear regulatory commission
LUT	look up table
ML	machine learning
NN	neural network
PINN	physics-informed neural network
PDE	partial differential equation
RF	random forest
SVR	support vector regression
MSE	mean squared error
RMSE	root mean squared error
RMSPE	root mean squared percentage error
MAPE	mean squared percentage error

Acknowledgments

Concludere questo percorso di studi è per me motivo di grande orgoglio, e vorrei dedicare alcune parole di ringraziamento a chi ha reso possibile questo traguardo.

In primo luogo, vorrei ringraziare il Professor Enrico Zio per l'opportunità di poter seguire un progetto così interessante e stimolante come lavoro di tesi e per la sua guida preziosa durante tutte le sue fasi. Un ringraziamento non da meno va al Professor Ibrahim Ahmed per aver seguito e controllato dall'inizio alla fine sia il lavoro che la stesura di questa tesi.

Un ringraziamento speciale ora va soprattutto alla mia famiglia, ai miei genitori che fin dai primi giorni di scuola non mi hanno mai fatto mancare una parola di sostegno o un abbraccio quando si presentava qualche ostacolo, piccolo o grande che fosse, e a mia sorella che è stata la prima persona a cui ho confidato tante difficoltà incontrate in questi anni. Grazie per avermi dato l'opportunità di poter decidere il mio percorso, anche quando le mie scelte non vi erano molto chiare, e di avermelo fatto affrontare senza altre preoccupazioni. Per questo sono stata molto fortunata. Restando in famiglia, i ringraziamenti più importanti vanno ai gatti di casa Nala e Minou. Nala, che non mi ha mai fatto mancare né il suo sguardo giudicante né una coccola serale e Minou, senza i cui sonnellini sul mio computer questa tesi sarebbe stata probabilmente completata mesi prima.

Vorrei anche esprimere la mia gratitudine ai miei amici, sia quelli che ho incontrato durante questi anni universitari sia quelli che mi accompagnano da sempre. Innanzitutto, Valentina Bilotti, quel primo giorno di liceo in cui ci siamo sedute vicine non sapevo quanto saresti diventata importante per me. Non so come sarei arrivata fin qui senza il tuo supporto, la tua positività e il tuo continuo voler cercare qualcosa di bello anche nelle giornate grigie. Ci sei sempre stata , questa laurea è un po' anche tua. Alice, Valentina, Benedetta C. e Margherita, vi conosco dall'asilo e ci siete state in ogni momento importante della mia vita e sono davvero grata che ci siamo trovate e soprattutto che siamo ancora qui tutte insieme.

Letizia e Benedetta B., voi siete entrate nella mia vita durante l'università invece ma siete presto diventate tra le mie persone preferite. Benni sei una delle persone migliori e più forti che io conosca, puoi anche trasferirti in Spagna ma sappi che non mi scappi. Leti, che si tratti di mangiare salame e fare gossip a fine serata, di una chiamata difficile a mezzanotte o di una giornata di studio al politecnico non hai mai detto di no, grazie.

Infine Riccardo, un piccolo ringraziamento va anche a te, quando ti chiedevo aiuto non me lo hai mai fatto mancare, anzi avresti anche voluto fare di più, e non hai mai avuto dubbi sul mio raggiungere questo risultato, avevi ragione e ora lo vedo anche io.

Questo traguardo è frutto di un cammino condiviso, e queste righe non bastano a esprimere la mia gratitudine per tutti coloro che, con il loro affetto e la loro fiducia, mi hanno accompagnato fino a qui. Se ho dimenticato qualcuno perdonatemi, sappiate che non vi ho dimenticati ma sono solo stata presa dall'emozione di poter scrivere finalmente queste righe.

Grazie di cuore a tutti.

Introduction

Two-phase gas-liquid flow systems exist in many engineering applications such as evaporators, boilers, air ejectors, turbines and steam generators. The technological importance of an accurate description of this type of flow is of paramount importance especially in the aerospace and nuclear industries, in which strict safety limits and extreme working conditions are present. Boiling heat transfer is defined as a mode of heat transfer that occurs with a phase change from liquid to vapor. Because of the high heat transfer rate which can be achieved, it has been used for applications requiring important cooling (as rocket motors and water-cooled nuclear reactors) [1]. However, if the bubbles population becomes too high, preventing the liquid from wetting the heated surface, a sharp deterioration in the heat transfer coefficient of the system occurs which marks the transition from nucleate boiling to film boiling. This situation results in system overheating which can lead to structural failure or even phase changes (and thus unpredicted behavior) of one or more components. This event is found in literature under the names "boiling crisis", "burnout" or "departure from nucleate boiling" (DNB) interchangeably, and the maximum heat flux achievable just before reaching such crisis is called critical heat flux (CHF). An accurate prediction of CHF is crucial for both safe and optimal performance [2].

CHF corresponds to the point where heat transfer is maximized, but it also marks the beginning of a rapid degradation of the heat transfer mechanism. As such, the goal is to operate as close to it as possible to maximize heat exchange and cooling, without risking to exceed it and potentially compromise system integrity. Specifically referring to the nuclear sector, the interest is in securing the integrity of fuel rods of water-cooled nuclear reactors that contain radioactive fission products. In PWRs during normal operation, the temperature and pressure conditions are maintained such that the coolant stays in a single-phase liquid state. Nevertheless, two-phase flow can be present during transients or accidents and for this reason it must be studied thoroughly in order to design the reactor to avoid proximity to CHF [2]. Given that the occurrence of CHF cannot be totally excluded, it's important to assess which rods may experience it through neutron flux distribution analysis. Once the rods more likely to encounter this condition and suffer rod damage are identified, the cladding temperature profile is included in the safety considerations. Among the controlling parameters, peak cladding temperature (PCT) is one of the safety parameters most relevant for the prevention of loss of coolant accidents (LOCA). The minimum departure from nucleate boiling ratio (MDNBR) – the minimum ratio of expected CHF and to actual operating local heat flux – defines various thermal margins and it is a

regulatory limit for the licensing of commercial PWRs worldwide. While the numerical value for MDNBR depends on individual reactor design and the CHF model employed, the U.S. Nuclear Regulatory Commission (NRC) mandates a minimum MDNBR of 1,3 [3]. Despite the importance of CHF prediction, the complex nature of two-phase flow interactions and a lack of general consensus on the triggering mechanism of CHF, have hindered the development of a comprehensive physical description [4]. This phenomena has been extensively researched over the last six decades and an extensive library of developed correlations can be found in literature, most of which can make reasonably good predictions in their range of applicability [5]. However, out of their range of applicability, the uncertainty highly increases and CHF models are usually used in reactor safety analysis codes by combining more than one correlation to cover all flow regimes of interest [6].

Based on flow quality there are two main types of CHF mechanisms: departure from nucleate boiling (DNB) and Dryout. The first predictive tools proposed by the scientific community were data-driven empirical correlations and physics-driven mechanistic models). Furthermore, when additional factors are present, such as transients, non-uniform heat flux distribution, rod bundles etc., the complexity increases significantly and thus, the need for a more universal CHF prediction method has led to the development of the 2006 Groeneveld look up table (LUT), a normalized databank built from extensive experimental testing results, making up for the limitations of the theoretical CHF modelling. Nevertheless, it does not reach the accuracy level needed and thus high safety margins are still required.

Recent advancements in artificial intelligence (AI) and machine learning (ML) allowed the investigation of innovative approaches to tackle complex phenomena like CHF. ML algorithms used for predicting CHF are usually supervised regression algorithms, which means that they use known CHF data for model training. Among the different types of ML models investigated, fully connected neural networks are easier to implement and show better generalization capabilities [7]. The main problem ML is facing it's that it requires minimal priori knowledge. This "black-box" nature is the main drawback for global acceptance [8] and it will be addressed in this work with ML explicability tools like LIME (Local Interpretable Model-agnostic Explanations), for local interpretability, and SHAP (SHapley Additive exPlanations), for global feature importance evaluation. A compromise between ML and physics has been found of late in the so called physics-informed neural networks (PINNs). They combine the power of neural networks (NNs) with the physics principles governing the phenomena under investigation by training the network on available data and incorporating physical laws in its learning process. In the nuclear field it has been used for example to solve point kinetics equations (PKEs) using a particular PINN framework called X-TFC (extreme theory of functional connections) [9]. The implementation of prior knowledge in machine learning algorithms has been successfully explored in various disciplines, such as aerospace engineering [10],

electrical engineering [11], materials science [12] and also CHF prediction [8] as well as general heat transfer problems [13], [14].

A nuclear reactor is a complex system made of multiple interconnected disciplines such as chemistry, nuclear physics, heat transfer, mechanics, fluid dynamics, radiation effects, electronics and so on. The design process is finalized at reaching maximum safety and optimal efficiency and economics under such conditions. To do so, the extensive amount of data gathered by the nuclear sector is an important starting point for the development of AI technologies to strengthen the field of nuclear energy. Many studies have investigated the possibility to integrate this tool for core design, radiation shielding, thermal-hydraulic analysis, operation and maintenance and so on [15].

This work proposes a systematic approach to develop a NNs ensemble that leverages domain-specific knowledge, integrated through LUTs, for predicting CHF. In addition, a PINN model is developed to investigate the potential benefits of incorporating CHF correlations into the ML framework developed for this type of problem. The existing literature predominantly features standalone CHF prediction models, although a few studies have experimented with hybrid or ensemble approaches to improve prediction accuracy. While combining multiple models has shown promising performance improvements, it often introduces two potential issues: 1) an overly complex ensemble model could improve performance but also lead to overfitting problems, and 2) a less complex ensemble model could not maximize the prediction performance of NNs. Therefore, developing and optimizing ensemble models by balancing predictive accuracy and model complexity is essential.

The proposed method is validated using experimental CHF data made available by the Working Party on Scientific Issues and Uncertainty Analysis of Reactor Systems (WPRS) Expert Group on Reactor Systems Multi-Physics (EGMUP) task force on AI and ML for Scientific Computing in Nuclear Engineering projects, promoted by the OECD/NEA [16]. The results demonstrate that the ensemble model outperforms individual models and other state-of-the-art methods, as well as PINN results which seems to be limited by the present physical understanding of the CHF.

1 Overview of Critical Heat Flux

There are two types of boiling: pool boiling and flow boiling. In the first one (quiescent liquid) buoyancy effects are dominant, in the latter forced-convection effects are prevailing. In Figure 1.1, pool boiling and flow boiling regimes are shown through the Nukiyama curve. Initially, single-phase natural convection is present as water is heated up. With increasing heat flux, the temperature gets high enough to activate nucleation sites and reach the "onset of nucleate boiling" point, initiating bubble formation at the wall. This marks the transition from free convection to nucleate boiling. In this region, an increase in wall superheating leads to increased bubbles production and movement producing higher heat transfer rates even with small temperature differences. Hence, the nucleate boiling region is the target operating region for thermal engineering applications.

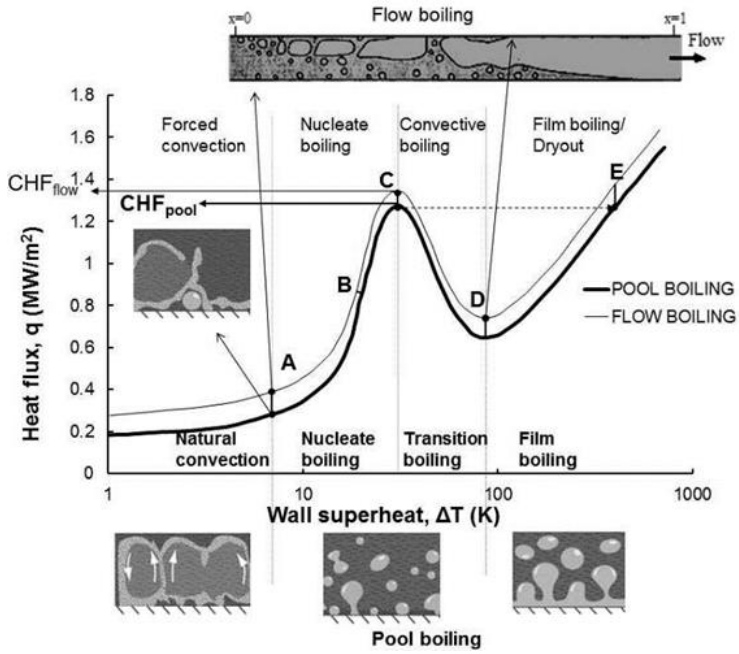


Figure 1.1: Nukiyama curve for pool and flow boiling [17]

At a certain point, the bubbles production is so high that they start to merge creating an insulating layer and thus degrading the heat transfer capabilities of the system due to the worst heat exchanging properties of vapor. This is known as "boiling crisis" and its starting point from the plot in Figure 1.1 is the one with maximum heat flux before the degradation process starts and it's called Critical Heat Flux (CHF). The following

region is called partial film boiling or transition boiling region. Here, an increase in wall temperature corresponds to greater vapor blankets and decreasing heat transfer properties.

When a stable vapor film is formed, the heat transfer reaches a minimum (point D, the Leidenfrost point) and the last part of the plot is entered, called film boiling region. In this region a radiative heat transfer contribute is present, which increases with wall temperature. However, it's not convenient to operate in this region because to exchange the same heat flux that can be exchanged in the nucleate boiling region, a much higher temperature is needed. Also, the last part of the plot presents a much smoother increase than that depicted in Figure 1.1, underlying the incredibly high temperatures that would be needed with respect to the nucleate boiling region ones. Taking into consideration also materials thermal limitations, this region is avoided in practical applications.

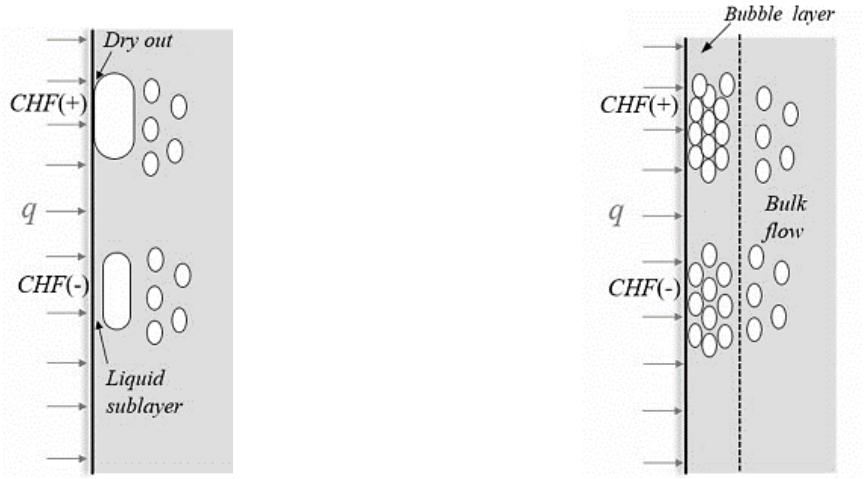
Up to now pool boiling has been described, but the same mechanisms apply to flow boiling, considering higher heat transfer coefficients due to the enhanced convective contribute of the two-phase flow. The deterioration of the heat transfer coefficient during a boiling crisis overheats and possibly damages the wall, which is particularly dangerous when considering a nuclear reactor fuel rods. For this reason, the nature of the boiling crisis must be fully understood to design and ensure the safety of water-cooled nuclear reactors. For maximum efficiency it's important to work close to the CHF region and for maximum safety it's essential to not encounter such thermal crisis under any operational or transient condition.

1.1. CHF Initiating Mechanisms

As previously mentioned, the nucleate boiling region is the one considered for engineering designs and the target is to work as close as possible to the CHF for maximum heat exchange capacity. Accurate CHF characterization depends on understanding the triggering mechanism of the boiling crisis event. Different theories have been explored and the boiling crisis can be found in literature also under the names of burnout or dryout (DO) for annular low, and departure from nucleate boiling (DNB) for bubbly flow, depending on the encountered bubble behavior.

Current literature on CHF models is divided over this initiating mechanism, nonetheless most researchers agree that flow conditions in the immediate vicinity of the heated surface are responsible for initiating CHF [5]. While other interpretations exist, the two descriptions more adopted are the DNB-type CHF in the subcooled or low-quality region, and the DO-type CHF in the high-quality region. Both mechanisms are explained in the following two sections.

1.1.1. Departure from Nucleate Boiling



(a) Liquid sublayer dryout (b) Near wall bubble crowding

Figure 1.2: DNB type of CHF mechanisms [18]

At low quality conditions, even subcooling ones, the boiling crisis known as DNB sets the limit, and nucleate boiling heat flux cannot be increased indefinitely. In this condition, a vapor blanket develops from bubble crowding near the heated wall. This sort of insulating blanket prevents the liquid from directly touching the wall. The heat transfer coefficient of vapor is much worse than that of a liquid, thus leading to a rise in the wall temperature. The DNB initiating mechanism has been described mainly following two different models:

- *Liquid sublayer dryout*: this model is first proposed for pool boiling assuming vapor blankets to be formed from small bubbles diverging into vapor jets. Then, the instability at the liquid-vapor interface, known as the Kelvin-Helmholtz instability, causes the jets to merge at some point and when the liquid trapped between the jets is fully vaporized, vapor blankets are formed. This work has been later extended to flow boiling in vertical cylinders [19] assuming bubbles to grow vertically, obstructing the liquid flow coming from the bulk as shown in Figure 1.2a. Once the liquid layer beneath the blanket fully dries out, CHF is set to occur.
- *Near wall bubble crowding*: this model is based on the presence of a bubble layer adjacent to the heated surface as shown in Figure 1.2b. The exchange between this layer and the bulk liquid appears to be limited. When the void fraction of the bubbly layer reaches a critical level, the liquid is almost completely prevented from directly contacting the wall.

1.1.2. Dryout

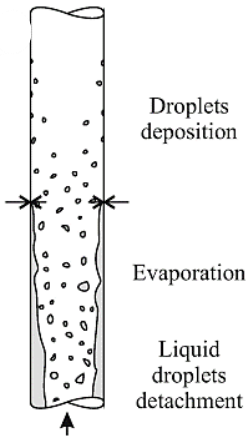


Figure 1.3: DO type of CHF [20]

In high quality regimes and annular flow conditions, DO is the primary phenomenon responsible for triggering CHF. Dryout is the result of the gradual disappearance of the liquid film adjacent to the wall because of both entrainment and evaporation as shown in Figure 1.3. Entrainment of liquid droplets by the vapor core can be induced by various mechanisms, including the formation of waves on the liquid film surface due to shear forces. These waves may release droplets into the vapor core. Additionally, bubbles forming within the liquid film can burst, dispersing liquid particles into the flow, which further contributes to the thinning of the liquid film.

This phenomenon is particularly significant in systems operating at high mass flux conditions, where the shear forces are further increased due to the high velocities, thereby enhancing entrainment. When the liquid film becomes sufficiently thin, it can no longer effectively wet the surface, which ultimately results in dryout.

1.2. CHF Prediction Methods

This section delves into the up to date methods for predicting CHF. Both empirical and mechanistic models are discussed, as well as more recent techniques such as look up tables (LUTs) and artificial intelligence-based models.

1.2.1. Empirical and Mechanistic Models

The predictive tools for CHF proposed during the past 70 years by the scientific community can be categorized in data-driven empirical correlations and physics-derived mechanistic models. Those derived experimentally are easy to implement and show good agreement when applied to similar operating conditions. However, they are data-fitting mathematical correlations and lack generalization beyond their range of validity. Mechanistic models, while incorporating aspects of the underlying physics such as flow pattern, bubble size and velocity, are also constrained, as the CHF

phenomenon is not yet well understood. Consequently, the level of accuracy required for water-cooled nuclear reactor design and safety analysis cannot be achieved with mechanistic models alone, often necessitating the use of empirical parameters derived from design-specific data [4] to be implemented in nuclear reactor codes.

Table 1.1: Summary of well-known CHF correlations

CHF References	Type	Described scenario
Tong [21], 1967	Empirical	DNB, Near wall bubble crowding
Biasi [22], 1968	Empirical	DNB, Near wall bubble crowding
Hughes [23], 1970	Empirical	DNB, Near wall bubble crowding
Bowring [24], 1973	Empirical	Dryout
Levitan [25], 1975	Empirical	DNB, Near wall bubble crowding
Whalley [26], 1977	Mechanistic	Dryout
Weisman & Pei [27], 1983	Mechanistic	DNB, Near wall bubble crowding
Lee & Mudawar [19], 1988	Mechanistic	DNB, Liquid sublayer dryout
Chang & Lee [28], 1989	Mechanistic	DNB, Near wall bubble crowding
Lin et al. [29], 1989	Mechanistic	DNB, Liquid sublayer dryout
Katto [30], 1992	Empirical	DNB, Liquid sublayer dryout
Celata et al. [31], 1999	Mechanistic	DNB, Liquid sublayer dryout
Kwon & Chang [32], 1999	Mechanistic	DNB, Near wall bubble crowding
Kataoka [33], 2000	Mechanistic	Dryout
Liu [34], 2022	Mechanistic	DNB, Liquid sublayer dryout

Table 1.1 presents a selection of commonly referenced CHF correlations. One of the earliest model for dryout was presented by Bowring from bundle data and it was later

extended by Whalley. Following their work, Kataoka presented a new correlation based on entrainment and deposition rate descriptions more aligned with experimental data and physical knowledge. Considering departure from nucleate boiling, Lee & Mudawar addressed the liquid sublayer dryout mechanism. Katto later extended the pressure validity range of the work of Lee & Mudawar and Lin strengthened it with more solid theoretical basis. Both models have a mechanistic foundation but they also make extensive use of empirical parameters. To reduce this dependency Celata et al. proposed a less data-dependent model. More recent attempts by Liu have shown a prediction accuracy of CHF of ±30% for both water and R134a for subcooled flow boiling. Tong first described the near wall bubble crowding mechanism and from this many subsequent works proposed their own formulations introducing additional features and modifications. The Westinghouse correlation was widely used in western countries, whereas the correlation of Levitan found wide application in the former Soviet Union [35].

The focus should now be on extending the experimental dataset available [5]. This would provide a more solid background from which to derive more accurate physical observations on the parameters governing bubble movement and vapor formation, leading to more precise and reliable mechanistic models. Current literature presents a great variety of correlations, however, for many engineering applications, more than one correlation is needed to cover all possible thermal-hydraulic conditions. For this reason, conceptual maps have been developed through the years considering possible models combinations depending on the flow regimes. The modified Barnett correlation by Hughes, Westinghouse-3 and the Biasi correlation are commonly used in the nuclear industry. A model combining all of them is shown to produce 75% of the predictions within 30% error with respect to experimental results [36].

1.2.2. Look Up Tables

A vast number of CHF models and correlations has been developed, each with its limited range of applicability. Due to this limitation, it's usually necessary to use more than one correlation to cover the whole range of conditions encountered in a thermal engineering system. A more generalized CHF prediction method was necessary.

The first attempt to construct a table of CHF values was made by Doroshchuk in 1975 [37], using a dataset of 5000 data points available at that time. This work has been continuously updated by incorporating new available experimental data. The most significant effort in this regard was led by Groeneveld et al. in 1995 [38] which comprised of about 24000 data points. Afterwards, between 1995 and 2006, 27 additional CHF datasets for vertical water flow in uniformly heated tubes were acquired, managed and added. This work culminated in the 2006 CHF LUT published by Groeneveld et al. [39], which is the largest endeavor to collect and correlate CHF values over a large parameter space. It's based on a dataset made of about 25,000 points and provides CHF values at 24 pressures, 20 mass fluxes and 23 qualities, covering

many conditions of practical interest for vertical water-cooled tubes. For any given triplets of P, G, X, linear interpolation is used to obtain CHF values at in-between table conditions. The 2006 Groeneveld CHF LUT is a normalized data bank with respect to tubes with diameters of 8mm and thus the following correction is performed to adjust the predicted CHF to the actual tube diameter [39]:

$$\mathrm{CHF_D} = \mathrm{CHF_{8mm}} \left(\frac{\mathrm{D}}{8}\right)^{-0.5} \tag{1.1}$$

By using the equivalent diameter in the above equation, the result is corrected for bundle geometries. The LUT presents many advantages compared to other prediction methods:

- Simple and of immediate practical use
- Increased number of experimental data in the dataset
- Improved CHF prediction accuracy
- Eliminates the need to choose among all CHF correlations depending on the considered situation

However, the 2006 LUT has some limitations, such as the lack of any physical knowledge and the scarcity of data in some parts of the domain such as high flows and high qualities, low flows and low qualities and pressure range 0.2 to 0.5 MPa. Applications in nuclear engineering face another limitation: operating conditions are difficult to simulate in order to collect a significant amount of experimental data (essential for reliable results), which are also needed to determine empirical parameters. For transient heat transfer in rod bundles, adjustment factors are introduced to correct for flux shape, transient effects and so on. Overall, the Groeneveld 2006 LUT has proven quite reliable and has been widely adopted [6].

1.2.3. Review of Artificial Intelligence Approaches

Previous works have already investigated the application of AI techniques for improving CHF prediction. One of the earliest approaches was proposed by Mazzola [40], who explored the use of artificial neural networks (ANNs) to predict empirical correlations parameters, traditionally obtained through best-fit techniques. Lee et al. [41] used a backpropagation neural network was used to predict the minimum DNBR for the study of a Total Loss of Flow Accident (TLOFA), demonstrating higher accuracy in prediction. Jiang and Zhao [42] developed a hybrid model combining support vector regression (ν-SVR) and radial basis function networks (RBFN), improving prediction accuracy for CHF in vertical round tubes compared to standard SVR and empirical correlations. In [43], the work on ν-SVR was extended considering sparsely distributed CHF data, showing better accuracy by training near critical inflection points. More recently, Atta Ullah [44] investigated the performance of

standalone ANNs, SVMs, and RF, finding that hybrid models combining ANNs and LUTs outperformed standalone models. Zhao and coworkers [8] also proposed a hybrid framework, integrating physics-informed machine learning with domain knowledge. This approach extended the applicability domain and improved performance and generalization capabilities over traditional models for various flow conditions. Also Mao and Jin [45] combined physics and ML by integrating physics models in the description resulting in physics-informed ML (PIML). Their study for predicting CHF showed that LUT-informed NN were the most stable and robust. While Khalid [46] considered an ensemble of deep sparse autoencoders (AEs) for feature extraction, coupled with a deep neural network (DNN) as a meta-learner for predicting CHF. This method addressed the limitations of existing models by leveraging a large dataset covering diverse operating conditions, significantly improving accuracy and reliability. In the nuclear field, physics-informed NN have also been previously used for other applications, for example, to solve point kinetics equations (PKEs) using a particular PINN framework called X-TFC (extreme theory of functional connections) [9].

2 Problem Formulation

The nuclear reactor lifecycle comprises of multiple steps, each with specific engineering challenges and safety considerations. Starting with the design phase, then into manufacturing and assembly, followed by operational phases, where refueling, maintenance and monitoring ensure continuous performance and safety. To close the cycle there is the decommissioning stage for dismantling and radioactive waste disposal. The present work sets itself in the design phase, where accurately predicting the CHF is crucial for ensuring safe operating conditions within water-cooled reactors. In this research, fully connected neural networks' architectures are leveraged for predicting CHF, while comparing their performance against traditional look up table methods. It is a heat transfer problem, referring to water-cooled reactors, described through a set of P most relevant physical variables, all measured at defined boundary conditions at a generic observational point i:

$$\mathbf{y_i} = \left[y_{i,1}, \dots, y_{i,j}, \dots, y_{i,P}\right] \tag{2.1}$$

These physical inputs include both geometric and hydraulic parameters, such as hydraulic or equivalent tube diameter (D), heated length (L), pressure (P), mass flux (G), outlet quality (X), inlet subcooling (Δh) and inlet temperature (T). From extensive literature review, these are the parameters most directly affecting the prediction of the CHF at the same observational point i, q_i^{CHF}. The inclusion of both inlet subcooling and inlet temperature is decided to allow the model to extract a wider amount of information from all available features, even though they must be considered redundant as they are related to the quality. This strategy aims at maximizing predictive accuracy by incorporating as much relevant data as possible, despite some potential overlap. It is considered the availability of a dataset $\mathbf{D} = \left[\mathbf{y_i} q_i^{CHF}\right]_{i=1,\dots,N}$ made of experimental results collected over a specific time period, which consists of:

- The measurement matrix $Y \in R^{NxP}$, which contains experimentally collected data. In this matrix, $y_{i,j}$ represents the measured physical quantity j at the observation point i, with i=1,…,N and j=1,…,P.
- The corresponding CHF vector $\mathbf{q}^{CHF} \in \mathbb{R}^N$, which contains the CHF measurements q_i^{CHF} at each observation point i, with i=1,…,N, with the various input conditions defined in the measurement matrix.

In this context, considering a new test input $\mathbf{y}_{\text{test}}$ measured at the current observation, the objective of this work is to develop a data-driven model that receives in input $\mathbf{y}_{\text{test}}$ and predicts the corresponding CHF value $\widehat{\mathbf{q}}_{\text{test}}^{\text{CHF}}$. It is important to note that the following conditions are implicitly satisfied:

- There exists an unknown function $\mathcal{F}: q^{CHF} = f(y)$, which maps the physical quantities y measured during experiments to the corresponding CHF q^{CHF}.
- The amount of data points N in the dataset D is sufficient to approximate the function $\mathcal{F}$ with satisfactory accuracy.

Given the impossibility of verifying these two conditions theoretically, the present work makes the empirical assumption that they hold if a predictive model, $\mathcal{F}^*: \hat{q}^{CHF} = f^*(y)$ can be developed, and if it can estimate the true value of q^{CHF} with satisfactory performance.

3 Proposed Methods

In the previous sections the CHF problem has been presented, alongside its importance and challenges for its accurate modelling description. The focus will now be on describing the inner mechanism and workflow of the selected machine learning approaches and how to implement them to overcome the CHF prediction challenges and enhance its estimation accuracy.

Given the complexity and non-linearity of the relationships between the physical input parameters and the CHF, NNs are particularly well-suited for this task due to their capability to model intricate patterns and dependencies within the data [8]. In Figure 3.1, it's shown the framework developed for the presented work for the CHF prediction using an optimized ensemble of NNs. It comprises of two phases: first, the construction of individual NN models, separately trained and tested, and second the optimization of an ensemble of these models. The initial phase, explained in Section 3.1.1, focuses on training different, individual neural networks, with distinct architectures and hyperparameters, in order to identify the best performing models. The second phase, presented in Section 3.1.2, optimizes each ensemble of these trained models, aggregating all considered predictions (input variables are passed through each individual model independently and then combined) to produce more accurate and reliable estimates of CHF.

Traditional supervised neural networks rely solely on available data for training, while physics-informed neural networks leverage known physics principles to guide the learning process. This approach, presented in Section 3.2 ensures that the model's predictions are consistent with the underlying physical laws describing the phenomena. By doing so, the network learns from the data and at the same time obeys the constraints imposed by the physical correlations. In this last section, the PINN scheme is discussed, considering the different options to integrate the Westinghouse correlation, presented in Section 3.2.1, in the loss function definition to evaluate the best approach, starting from the same NN architectures developed in the first part and then refined. Finally, in Section 3.3 the performance metrics used throughout this study is presented and discussed.

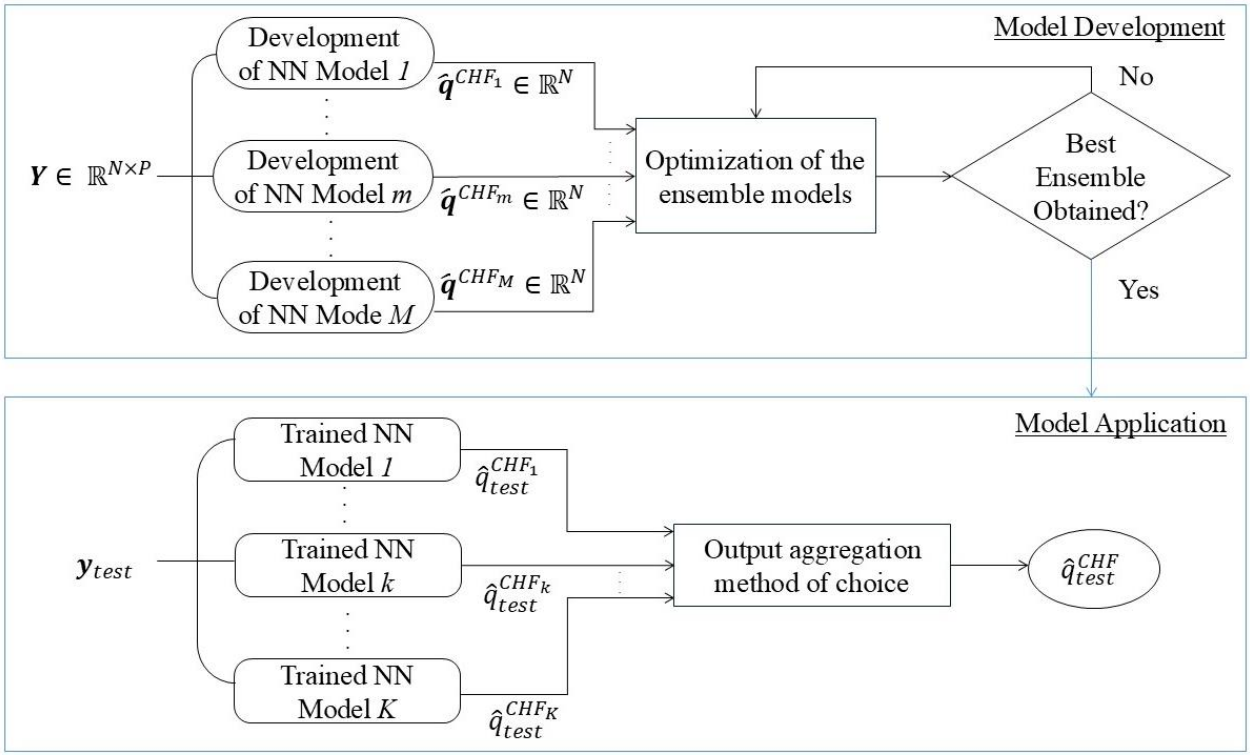


Figure 3.1: Flow diagram of proposed ensemble model for CHF prediction

3.1. Proposed Method 1: Ensemble of Neural Networks

3.1.1. Individual Neural Networks

The history of neural networks dates back to 1943 by Warren McCullough and Walter Pitts [47] and through the years NNs have been at the center of both neural and computer science. Today, they are widely used in various applications such as image and speech recognition [48], data classification and clustering at high velocity [49]. Deep learning and neural networks tend to be used interchangeably. However, the term "deep" only refers to the depth of layers in a neural network. Usually, a NN with more than three layers can be considered a deep learning algorithm, otherwise it's just a regular neural network.

Neural networks were developed taking as reference the working mechanism of the human brain. Their architecture simulates the way neurons work together to analyze a problem, make decisions and arrive at a conclusion starting from input informations. A NN is made of layers of nodes (the equivalent of brain neurons) organized in an input layer (which receives the input data vector $\mathbf{y}_i$), one or more hidden layers, and an output layer (providing the final prediction of CHF, $\hat{q}_i^{CHF}$). Figure 3.2 shows a typical representation of a NN model. The number of hidden layers and nodes within each layer determines the model's capacity to capture complex patterns in the data. The input layer takes as many nodes as the number of input parameters [47], which

are then connected to the nodes of the next layer through an activation function and their own associated weights and biases. The weights set the importance of the variable passing through that specific node in computing the output, while the bias acts as a sort of threshold for going forward or not.

If each node of a layer is connected to every node of the previous and subsequent layer it's called a fully connected neural network. However, this is not the only option, there are for example convolutional neural networks (CNNs) and recurrent neural networks (RNNs). The first is more commonly used for image classification problems and the nodes are connected only to a localized region of the previous layer. The second is applied to language processing and speech recognition and the connections between nodes are more sequence-based.

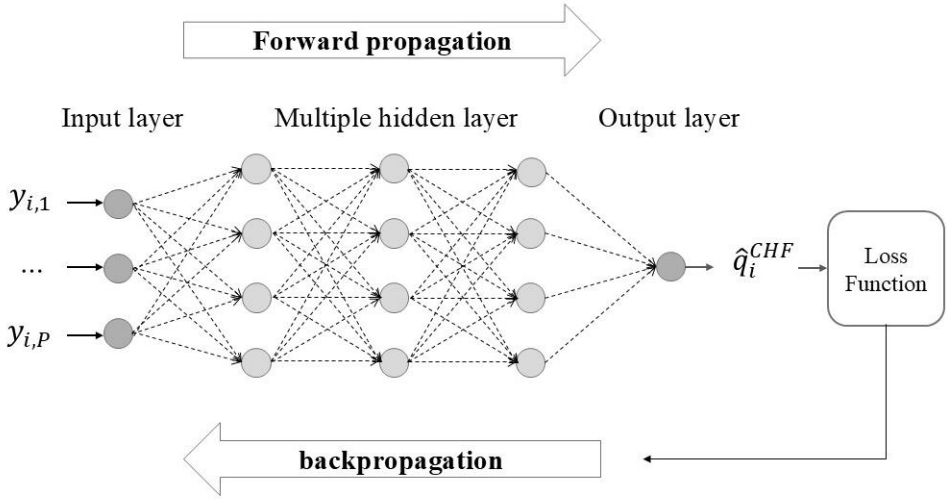


Figure 3.2: A typical NN architecture

Mathematically, a NN can be represented as a series of transformations applied to input data, whose development consists of three processes: forward propagation, loss function evaluation and backpropagation. In the forward propagation, the input is passed through the network in which, for each layer l, the output is computed as follows:

$$\mathrm{y}^{\mathrm{l}+1} = \mathrm{f}\left(\mathbf{W}^{\mathrm{l}}\mathrm{y}^{\mathrm{l}} + \mathbf{b}^{\mathrm{l}}\right) \tag{3.1}$$

where the inputs y to layer l are processed through an activation function f, which uses the weight matrix $\mathbf{W}^{\mathrm{l}}$ and bias vector $\mathbf{b}^{\mathrm{l}}$ of that layer, to obtain the output which serves as input for the next layer l+1 [50]. Common types of activation functions include ReLU (Rectified Linear Unit), sigmoid and hyperbolic tangent. Each function has unique properties that make them suitable for different types of neural networks. Finally, the target CHF prediction is computed by:

$$\hat{q}_i^{CHF} = f(\mathbf{W}^{L-1} y^{L-1} + \mathbf{b}^{L-1}) \tag{3.2}$$

where L is the total number of layers, excluding the input layer. By using supervised learning, the actual expected output can be used to adjust and fit properly the training parameters (weights and biases) for more accurate predictions, by means of backpropagation algorithms and optimization. During the training phase, weights and biases are iteratively updated to minimize a defined loss function (or cost function) representing the difference between actual q_i^{CHF} and predicted $\hat{q}_i^{CHF}$ output [51]. Generally, the mean squared error, as defined in equation (3.3), is used as the loss function for the training dataset used for the regression problem.

$$\mathcal{L} = MSE = \frac{1}{N}\sum_{i=1}^{N}\left(q_i^{CHF} - \hat{q}_i^{CHF}\right)^2 \tag{3.3}$$

This optimization procedure is usually performed through gradient descent, a first order iterative optimization algorithm. The calculations start with the gradient of the loss function with respect to the weights and bias, calculated with the chain rule, one layer at a time, in order to identify the direction and magnitude of the adjustments to make. Having chosen gradient descent as optimization algorithm, weights and biases are changed proportionally to the negative of the gradient of the loss function with respect to the "old" weights and biases according to the below equations. The learning rate α is the hyperparameter controlling how much weights and biases are adjusted at each iteration, until convergence, which is reached when the loss function in minimized.

$$w_{i,new} = w_{i,old} - \left(\alpha \frac{\partial L}{\partial w_{i,old}}\right) \tag{3.1}$$

$$b_{new} = b_{old} - \left(\alpha \frac{\partial L}{\partial b_{old}}\right) \tag{3.2}$$

The described NN model is used to develop multiple individual models, each trained separately with distinct architectures and hyperparameters settings, which are optimized following the procedure presented in Figure 3.3. Specifically, M different NN models are developed and trained independently during the training phase (Figure 1, top) using the measurements $\mathbf{Y} \in \mathbb{R}^{N\times P}$ as input data and $\mathbf{q}^{CHF} \in \mathbb{R}^{N}$ as output. Given the inherent stochastic nature of NNs due to random initialization of layer weights, training the same model architecture on identical data can yield multiple optimized models, each performing differently. To limit this randomness, and provide consistent results, a weight initialization technique is assigned in this work to each layer, so that only the hyperparameters are influencing the different

prediction results. To identify all potentially optimal NN models, the sequential procedure, summarized in Figure 3.3, is implemented as follows:

1) Develop and optimize NN models with L number of hidden layers through cross validation and varying model's hyperparameters such as batch size and learning rate;
2) Based on root mean squared error (RMSE) results from cross validation, select optimal NN models and optimize the number of nodes;
3) Compare the performance of the fully optimized NN models with L layers to that of the optimized models obtained with L – 1 layers;
4) If any model with L layers outperforms any of those with L – 1 layers, increment the number of hidden layers by 1 and repeat steps 1) o 3). Otherwise, proceed to step 5);
5) Collect all M finalized individual models to be used later for the ensemble model optimization.

Finally, each of the collected trained models generates its own CHF predictions $\widehat{\mathbf{q}}^{\mathrm{CHF}} \in \mathbb{R}^{\mathrm{N}}$ that will be used, as presented in the following paragraph, for the optimization of the ensemble model.

However, practical applications of neural networks face their own set of challenges, including the need of large, high-quality datasets for training, as well as the inherent difficulty in interpreting the internal mechanisms of neural networks (”black box" frame). It's therefore essential to integrate statistical measures and validating steps in the developing phase of the NN model. Both procedures will be presented in full in the following chapters.

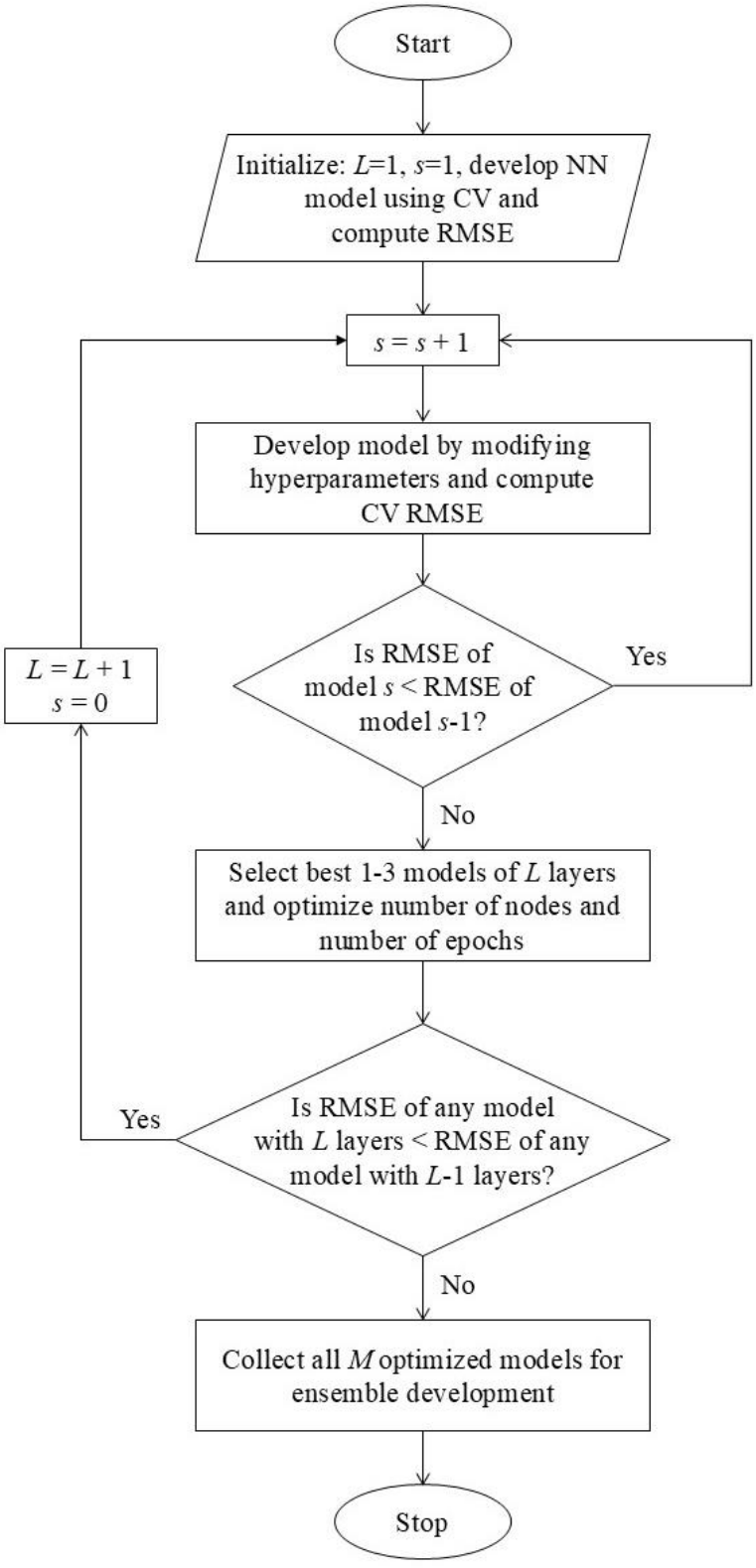


Figure 3.3: Flowchart of the development of multiple individual NN models

3.1.2. Optimization of Ensemble Model

Following the development and training of individual models, the next step is to group their collective predictions through ensemble learning, a technique that combines the output of multiple models to improve robustness and accuracy. A study by Hansen and Salamon [52] demonstrated that different models capture different data characteristics and combining them can mitigate the limitations of the single models, resulting in a more generalized solution and reducing overfitting. To identify the optimal ensemble of the developed NN models, the results from Section 3.1 are used to investigate the selection of the best combination of models among the *M* single models with minimal computational load (and without employing specific optimization algorithms), the combinatorics theory [53] is applied. Specifically, the proposed systematic approach is based on the following steps.

Step 1 – Calculation of potential ensemble models: the total number of possible combinations of two or more models from the *M* individual models is determined:

$$N^e = \sum_{r=2}^{M} C(M, r) \tag{3.3}$$

where $C(\cdot)$ is the binomial coefficient, i.e., the number of possible combinations of r models (r = 2,3,…,M) from M distinct models without regard to order and without replacement.

Step 2 – Enumeration model combinations: all N^e possible combinations of the M models are generated. For example, if M = 3, then from Eq.(5), $N^e = 4$ and the following potential combinations are produced: {1,2},{1,3},{2,3},{1,2,3}.

Step 3 – Evaluation of ensemble performance: the performance metrics (Chapter 4) for each ensemble configuration are calculated using the following aggregation methods:

- Mean aggregation:

$$\hat{q}_i^{CHF_{mean}} = \frac{1}{K}\sum_{k=1}^{K} \hat{q}_i^{CHF_k} \tag{3.4}$$

  where k is the number of models in the ensemble. This method provides a balanced prediction by averaging the over- and under- predictions.

- Median aggregation:

$$\hat{q}_i^{CHF_{median}} = \begin{cases} \hat{q}_i^{CHF_{\left[\frac{K+1}{2}\right]}} & \text{if K is odd} \\ \dfrac{\hat{q}_i^{CHF_{\left[\frac{K}{2}\right]}} + \hat{q}_i^{CHF_{\left[\frac{K}{2}+1\right]}}}{2} & \text{if K is even} \end{cases} \tag{3.5}$$

the median aggregation is less sensitive to outliers and provides greater resilience against extreme predictions.

Both methods are tested to determine the optimal balance between robustness and accuracy for CHF predictions.

Step 4 – Selection of the optimal ensemble: the ensemble that achieved the best performance based on the metrics computed in Step 3 is selected.

The proposed systematic approach not only enables the identification of the optimal ensemble but also reduces the computational demands associated with conventional optimization techniques, such as genetic algorithms or random search thereby enhancing efficiency in determining the ideal ensemble configuration.

3.2. Proposed Method 2: Physics-Informed Neural Network

Starting from the same foundational concepts presented in the previous paragraph, Raissi et al. [54] presented the so called physics-informed neural network technique. PINNs were originally developed to tackle two classes of problems: data-driven solution and data-driven discovery of partial differential equations. The interest is now on the first class of problems using gradient based optimization algorithms. Despite no guarantee on reaching convergence to a global minimum of the loss function, empirical evidence [54] indicates that, if the problem is well-posed and the solution is unique, this method can achieve good prediction accuracy with a sufficient number of data points and a sufficiently accurate neural network.

PINNs are neural networks that incorporate the phenomena governing equations, expressed as partial differential equations (PDEs), in the NN loss function as shown in Figure 3.4. PINNs can be used to solve PDEs, fractional equations, integral-differential equations and stochastic PDEs. Lots of researches can be found in literature, especially in recent years, for example for solving incompressible Navier-Stokes equations [55], and finding a solution for heat transfer [14] and solid mechanics [56] problems.

To describe in detail how PINNs work, let's consider the problem of solving the heat equation as presented in equation (3.6). It's a PDE describing how heat diffuses through a medium over time. The equation states that the rate of change of temperature u(x,t) with respect to time and at a given position is proportional to the Laplacian of u(x,t) through the thermal diffusivity coefficient α of the material.

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u \tag{3.6}$$

To solve this PDE with PINN, means to find the solution u(x,t). Let's indicate with u(x,t; θ) the model's output temperature prediction with the neural network defined

with model parameters $\theta = [W, b]$ and inputs x and t. By using automatic differentiation in PyTorch, the derivatives of the model output with respect to x and t are calculated. These derivatives approximate the actual partial derivatives of the PDE, as shown in equations (3.7) and (3.8). Other methods for computing derivatives in computer programs are manually evaluating the derivatives and coding them, numerical differentiation using finite difference approximations and symbolic differentiation using computer algebra systems. Manual differentiation is time consuming and prone to errors; numerical differentiation is simple to implement but highly inaccurate and scales poorly for gradients, making it unsuitable for machine learning; symbolic differentiation often results in complex and limiting results [57]. Automatic differentiation has been increasingly used in machine learning due to its ability, through the implementation of the chain rule, to avoid the numerical errors associated with the finite differences approach. Also, it supports high order derivatives, it's computationally efficient and it's easy to implement.

$$\frac{\partial u}{\partial t} = \frac{\partial u(x, t; \theta)}{\partial t} \tag{3.7}$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 u(x, t; \theta)}{\partial x^2} \tag{3.8}$$

The derivatives obtained though automatic differentiation approximate the real derivatives as accurately as the neural network correctly approximates the true solution of the PDE. These derivatives are used to construct the PDE residual and incorporate it into the loss function to guide the NN's training toward better modelling capacities given that the physical constraints are implemented in it.

The loss term associated to the PDE residual, and thus physics-related, is defined as:

$$\mathcal{L}_{PDE} = MSE = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\partial u(x_i, t_i)}{\partial t} - \alpha\nabla^2 u(x_i, t_i)\right)^2 \tag{3.9}$$

The loss term associated to the experimental data available is defined as:

$$\mathcal{L}_{data} = MSE = \frac{1}{N}\sum_{i=1}^{N}(u(x_i, t_i; \theta) - u_{true}(x_i, t_i))^2 \tag{3.10}$$

The loss function is defined weighting those two contributions by means of another hyperparameter λ to be optimized through validation and it is defined as follows:

$$\mathcal{L}_{total} = (1 - \lambda)\mathcal{L}_{data} + \lambda\mathcal{L}_{physics} \tag{3.11}$$

After training, which proceeds exactly as for neural networks, the system has learnt how to approximate the PDE solution across the entire domain. With this approach,

complex PDEs can be solved without the need of meshes or grids, including experimental data and physical correlations constraints at the same time. Finally, it must be noted that while PINNs are more often associated with dynamic systems described by differential equations, in this work it is applied to a static model description, the Westighouse-3 correlation as described in Equation (3.17). In the following paragraph it will be explained how this issue will be addressed in this work.

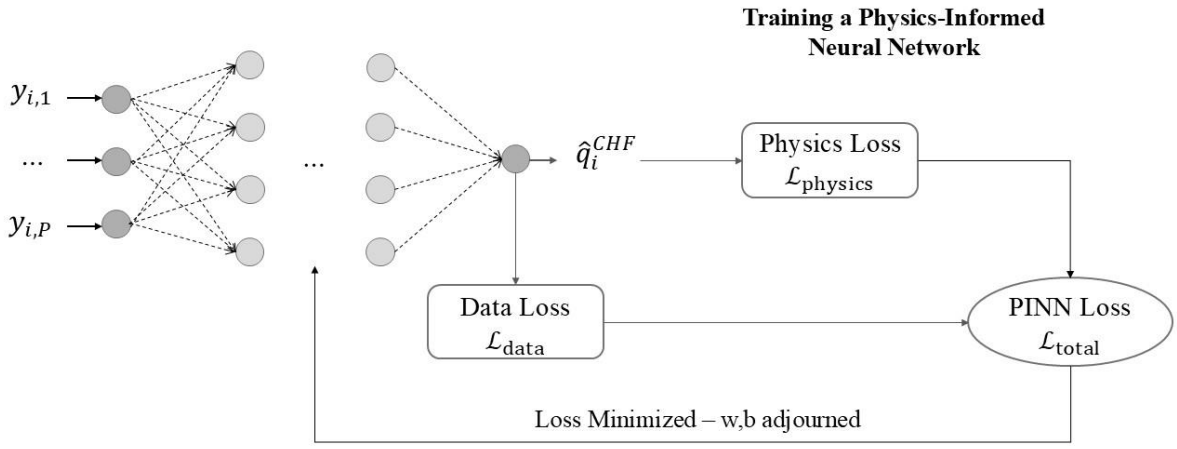


Figure 3.4: PINN schematic representation

3.2.1. Westinghouse W-3 Correlation

Given the large applicability in the nuclear field and the fact that it's easy to derive, the Westinghouse correlation has been selected for the implementation of the PINN and as such it is now presented in detail. Tong [21] developed an empirical correlation for predicting DNB for flow boiling conditions in a nuclear reactor design by considering both local DNB heat flux and location as important informations to be included in the description. The location determines the magnitude of the DNB heat flux and thus the temperature distribution at the tube wall after the boiling crisis. Then, the local DNB heat flux is necessary to evaluate the DNB ratio between predicted and reactor local heat flux to set the related core design criterion.

The work started by considering the local pressure, mass flux and thermodynamic quality as the main parameters affecting the flow pattern and hence the CHF. The system pressure determines the saturation temperature and the related thermal properties which in turn determine bubble size, buoyancy effects and so on. Coupled with local enthalpy, the local subcooling for bubble formation was included. By fitting the experimental data available at the time, Tong correlated P-G-X coupled relationships along with the local enthalpy and the diameter as geometrical factor. Each correlating function was defined by plotting an independent parameter at a time against measured CHF data, with all other parameters held at constat values [1]. The result expressed the heat flux as the product of all those factors:

$$q''_{crit} = F(X,P) \cdot F(X,G) \cdot F(D) \cdot F(H_{in}) \tag{3.12}$$

By explicitly writing all factors in equation (3.12), the following correlation, known as the Westinghouse W-3 correlation, was obtained:

$$\begin{aligned} \frac{q''_{DNB}}{10^6} = {} & \left[(2.022 - 0.00043P) + (0.172 - 0.000098P)e^{(18.177-0.004P)X}\right] \\ & \cdot \left[(0.148 - 1.596X + 0.173X|X|)G/10^6 + 1.037\right] \\ & \cdot \left[1.157 - 0.869X\right] \cdot \left[0.266 + 0.836e^{-3.151D}\right] \\ & \cdot \left[0.826 + 0.00079(H_{sat} - H_{in})\right] \end{aligned} \tag{3.13}$$

The unit of measure of the heat flux in this formulation is in Btu/hft² and the ranges of the parameters used in the correlation, as well as their units of measure, are:

$P = 1000 \div 2300$ psia

$G = 1.0 \cdot 10^6 \div 5.0 \cdot 10^6$ lb/hft²

$D = 0.2 \div 0.7$ in

$L = 10 \div 144$ in

$X_{loc} = -0.15 \div +0.15$

$H_{in} \geq 400$ Btu/lb

While the W-3 correlation was based on tests with flow inside vertical tubes, it also showed good agreement with flow outside fuel rod bundles by using the hydraulic diameter instead of the actual tube diameter. An additional factor could also be included to account for the mixing effects of grid spacers. Moreover, the W-3 correlation as defined in equation (3.13) can be applied only to axially uniform heat flux distributions. To account for non-uniform distributions, as it occurs with nuclear reactors' heat fluxes, a shape correction factor F is applied:

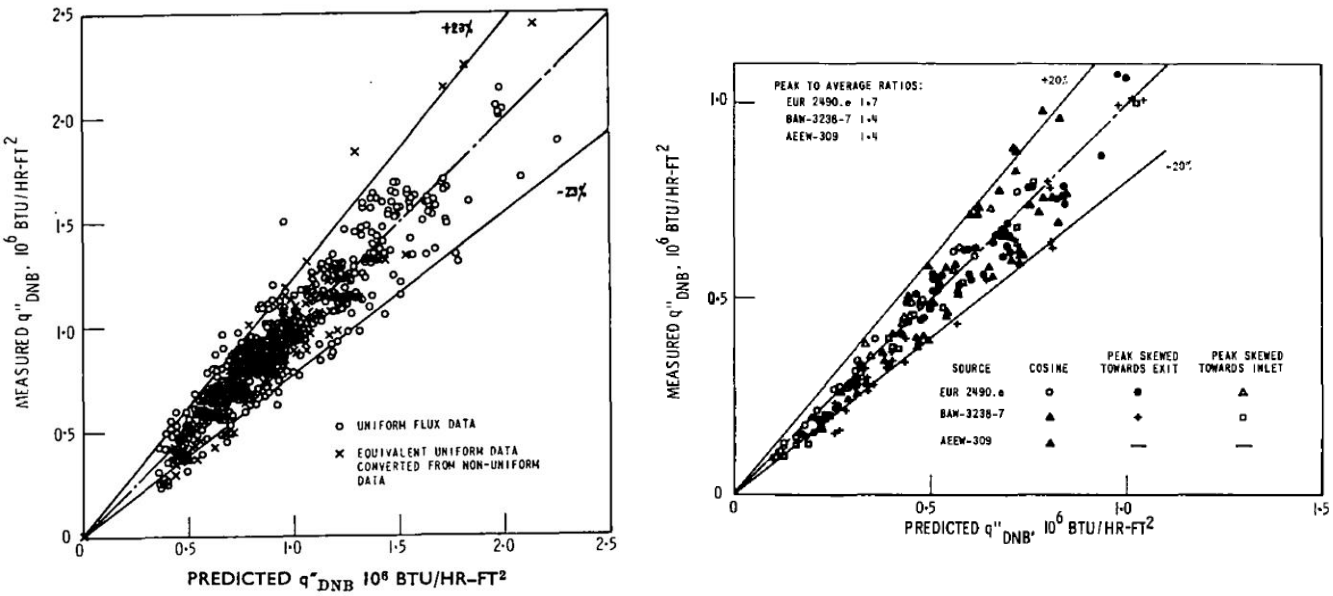
$$q''_{DNB,nu} = q''_{DNB,u}/F \tag{3.14}$$

where

$$F = \frac{C}{q''_{loc}\left[1 - \exp\left(-Cl_{DNB,u}\right)\right]} \int_0^{l_{DNB,nu}} q''(z)\exp\left[-C(l_{DNB,nu} - z)\right]dz \tag{3.15}$$

$$C = 0.15\frac{(1 - X_{DNB})^{4.31}}{(G/10^6)^{0.48}} \tag{3.16}$$

In the subcooled region the F factor is close to unity, while at higher qualities it becomes more relevant. In Figure 3.5 the results collected by Tong are presented for both types of heat flux distributions and an average 20% error boundary.



(a) Uniform W-3 (b) Non-uniform W-3

Figure 3.5: W-3 predicted vs measured CHF by Tong [21]

Considering the units of measure of the SI, the parameters validity ranges have been converted in the following values: 6.89÷15.86 MPa for pressure, 1356÷6781 kg/s/m² for mass flux, 0.005÷0.018 m for diameter, 0.25÷3.66 m for heated length and ≥930 kJ/kg for inlet enthalpy. Also, equation (3.17) shows the W-3 equation with adjusted coefficients to account for the change to units of measure of the SI, allowing for direct use of currently available experimental datasets.

$$\begin{aligned} q''_{DNB} = & \left[(2.022 - 0.062P) + (0.172 - 0.014P)e^{(18.177-0.599P)X}\right] \\ & \cdot [(0.148 - 1.596X + 0.173X|X|)2.334G + 3271] \\ & \cdot [1.157 - 0.869X] \cdot [0.266 + 0.836e^{-124D}] \\ & \cdot [0.826 + 0.00034(H_{sat} - H_{in})] \end{aligned} \tag{3.17}$$

When employing empirical correlations, such as this one, to determine operating conditions and safety margins for nuclear reactors, high conservative boundaries are implemented in order to guarantee safe operation under all circumstances [5]. Minimizing this extreme conservatism through AI innovative approaches can provide great economic advantage in terms of monetary savings and increased productivity through the maximum exploitation of the available resources

3.2.2. Definition of PINN: Loss Function

Considering that the considered CHF problem described through the Westinghouse-3 correlation is not addressed with PDEs, two different approaches for the loss function definition are examined. Both approaches are based on the presence of a data loss,

which is defined in the same way for both methods, and a physics loss: $\mathcal{L}_{\text{data}}$, as the data-driven loss function as in equation (3.10) and $\mathcal{L}_{\text{physics}}$, as the physical loss function. For consistency with the previous work, the data loss is the MSE of the residual between actual and model's prediction of the CHF. The two approaches for defining the total loss function differ only on the physics loss contribute:

- The first approach relies on the simple difference between the model's predicted CHF and the one calculated with the selected correlation. This makes sure the learning process goes in the direction of approximating the given physical correlation as accurately as possible. $\mathcal{L}_{\text{physics}}$ is defined as:

$$\mathcal{L}_{\text{physics}} = \frac{1}{N}\sum_{i=1}^{N}(\hat{q}_i^{CHF} - q_i^{Corr}) \tag{3.18}$$

- The second approach incorporates the partial derivatives, calculated with automatic differentiation algorithms, of the model and correlation's outputs with respect to each input to insert the local behavior of the CHF with respect to each variable. It's defined through the sum of the residuals for each input's variation as described below:

$$\mathcal{L}_{\text{physics}} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{L}_{D,i} + \mathcal{L}_{P,i} + \mathcal{L}_{G,i} + \mathcal{L}_{X,i}\right) \tag{3.19}$$

with

$$\begin{cases} \mathcal{L}_D = \left(\dfrac{\partial \hat{q}_i^{CHF}}{\partial D} - f'_D(D,P,G,X)\right)^2 \\ \mathcal{L}_P = \left(\dfrac{\partial \hat{q}_i^{CHF}}{\partial P} - f'_P(D,P,G,X)\right)^2 \\ \mathcal{L}_G = \left(\dfrac{\partial \hat{q}_i^{CHF}}{\partial G} - f'_G(D,P,G,X)\right)^2 \\ \mathcal{L}_X = \left(\dfrac{\partial \hat{q}_i^{CHF}}{\partial X} - f'_X(D,P,G,X)\right)^2 \end{cases} \tag{3.20}$$

To apply this procedure, the NN receives scaled input values, but the correlation should be evaluated using unscaled values. The scaling process will be addressed more specifically in the following Chapter. Then, for consistency reasons for the residual computation, the results from the correlation must be rescaled coherently

with the rest of the data. The developed NN works with tensors arrays and matrixes but "MinMaxScaler" (a class in scikit-learn designed for scaling) works on numpy arrays and as such, by applying the inverse_transform() step before going through the W-3 correlation, the computational graph is broken and the code is not able to evaluate the gradients during backpropagation.

Another estimator based on the same principles of "MinMaxScaler" is "minmax_scale", this function returns a scaled array of the same type of the input array. However, it does not allow to save the scaling parameters to be used for other arrays. This is an important step given that both validation and training sets must be scaled accordingly in two different moments. This evaluation led to the definition of a custom scaling function, which is presented in full in Appendix B.1 under the name of "MinMaxTensor". This scaling function has the same mathematical structure of "MinMaxScaler" but takes unscaled tensors as input and returns scaled tensors as output. Also, unlike "minmax_scale", it allows to retain the scaling parameters based on one array/matrix to be used for a secondary array/matrix.

3.3. Performance Metrics

In this section, the evaluation metrics used throughout this study are presented. A combination of more than one statistical measure is often required to comprehensively assess model performance. The following metrics, commonly used in the nuclear engineering sector [16], are employed to assess the performance of the proposed method on the test dataset $\mathbf{D}_{\text{test}} = \left[\mathbf{y_i}\, q_i^{CHF}\right]_{i=1,\dots,N^t}$ where N^t is the amount of datapoints in $\mathbf{D}_{\text{test}}$:

- The root mean squared error (RMSE) is computed as:

$$\text{RMSE} = \sqrt{\frac{1}{N^t}\sum_{i=1}^{N^t}\left(\hat{q}_i^{CHF} - q_i^{CHF}\right)^2} \tag{3.21}$$

- The root mean squared percentage error (RMSPE) is calculated as:

$$\text{RMSPE} = 100\sqrt{\frac{1}{N^t}\sum_{i=1}^{N^t}\left(\frac{\hat{q}_i^{CHF} - q_i^{CHF}}{q_i^{CHF}}\right)^2} \tag{3.22}$$

- The mean absolute percentage error (MAPE) is computed as:

$$\text{MAPE} = \frac{100}{N^t}\sum_{i=1}^{N^t}\left|\frac{\hat{q}_i^{CHF} - q_i^{CHF}}{q_i^{CHF}}\right| \tag{3.23}$$

- The Q^2-error is calculated as:

$$Q^2 = \frac{\sum_{i=1}^{N^t}\left(\hat{q}_i^{CHF} - q_i^{CHF}\right)^2}{\sum_{i=1}^{N^t}\left(\hat{q}_i^{CHF} - \hat{\mu}\right)^2} \tag{3.24}$$

The RMSE and its percentage RMSPE have the advantage of being on the same scale of the data with respect to the mean squared error. RMSE measures the absolute error, while RMSPE provides a measure of the relative error as a percentage of the actual/measured value, allowing comparison of prediction accuracy between different scales of measured values. This is particularly useful for large datasets where variability in data scale exists. However, RMSPE is more sensitive to outliers, penalizing larger errors more.

The MAPE provides a straightforward interpretation, as it directly represents the average percentage error across all predictions. This metric is particularly intuitive because it expresses the error as a percentage, making it easy to understand the magnitude of the error in relation to the actual values. The Q^2, also known as the coefficient of determination, evaluates how well a regression model's prediction reflects actual data values. It is a normalized error metric of the prediction error by the prediction deviation from the mean. It provides a relative assessment of prediction accuracy, considering the variability in the data. Values close to 0 indicate strong predictive capabilities of the NN model, while values greater than 1 suggest worse predictive capabilities than simply using mean value. Lastly, the amount of data points falling within a specific error threshold was derived from the residuals (r) and error percentages ($E_{\%}$) calculated as follows:

$$r = q_i^{CHF} - \hat{q}_i^{CHF} \tag{3.25}$$

$$E_{\%} = 100\left|\frac{r}{q_i^{CHF}}\right| \tag{3.26}$$

From equation (3.26), the error percentage of each row of predicted CHF is evaluated with respect to the actual value. Then, the number of rows with error lower than a specific percentage value (in this case 10 and 20) are counted and divided by the total number of rows multiplied by 100 to get the percentage. In Appendix A.2 it's displayed how the evaluation presented in this paragraph has been translated in the pertaining sections of the python code. After this characterization, hyperparameters optimization and baseline model definition can be addressed. For this type of problem, more than one model with different hyperparameters, number of nodes and hidden layer is considered and the performance of the single best model is later compared to ensemble models results, ultimately selecting the best approach. All results are presented in the following Chapter with particular attention to avoiding overfitting and ensuring good generalization to unseen data.

4 Case Study

The dataset used for this case study is now presented, from the considered parameters and their numerical values to the preprocessing steps performed on it before actually employing it in the PyTorch code for the model development. This dataset will be divided in two subsets, one used for validation and training, and the other for testing.

4.1. Dataset

To validate, train and test the proposed CHF prediction models, the 2006 Groeneveld look up table, published by the US Nuclear Regulatory Commission [6], is used. The NRC CHF dataset from Groeneveld is the largest CHF dataset publicly available. This availability provides the opportunity to explore supervised ML as predictive tool for CHF. The potential of ML has been demonstrated in ML regression analysis of previous works using this same dataset [7]. This dataset is made of 24579 points from 59 different sources, all documented in [6] by Groeneveld, where each point corresponds to a CHF measure in a vertical water-cooled uniformly heated tube under varying experimental conditions. Most measurements were obtained using thermocouples to detect CHF. Table 4.1 shows the parameter space of the boundary conditions and geometrical variables collected for all datapoints comprising the NRC CHF dataset. Those points serve as input parameters for the network construction and they can be classified as: geometric (tube diameter D, heated length L), measured (pressure P, mass flux G, inlet temperature T) and calculated parameters (outlet quality X, inlet subcooling H_{sub}). The calculated parameters were derived using water tables at saturation.

Table 4.1: Parameters range for NRC CHF dataset

	Tube Diameter	Heated Length	Pressure	Mass Flux	Outlet Quality	Inlet Subcooling	Inlet Temperature
	[m]	[m]	[kPa]	[kg/s/m^2]	[-]	[kJ/kg]	[°C]
Min	0.002	0.05	100	8.2	-0.497	-1211	5.89
Max	0.016	20	20000	7964	0.999	1644	361.94

Figure 4.2 shows a scatter plot matrix of the NRC CHF dataset points to get an overview of the ranges with large amount of data availability (and thus providing

higher reliability when modelled) and those lacking experimental points. Also, in Figure 4.1 are shown scatter plots evidencing the relationships between CHF and each input parameter. This allows for the evaluation of potential relationships between variables and the assessment of data distribution, as well as the study of possible input-output relationships.

For the analysis presented in this work, all input parameters are considered as inputs for the training, validation and testing of the best neural network framework. While, for the development of the physics-informed neural network, only 5 input parameters are used, namely D, L, P, G, X. This selection is done based on the fact that most CHF prediction models are currently function of those physical values. The other parameters, based on saturated fluid properties can be inserted, for example, to extend the model application to different fluids. From a ML point of view, having an excessively large number of input features can increase the risk of overfitting, as the model may capture noise and irrelevant patterns in the training dataset, leading to reduced generalization performance on unseen data. Also this effect is thus investigated.

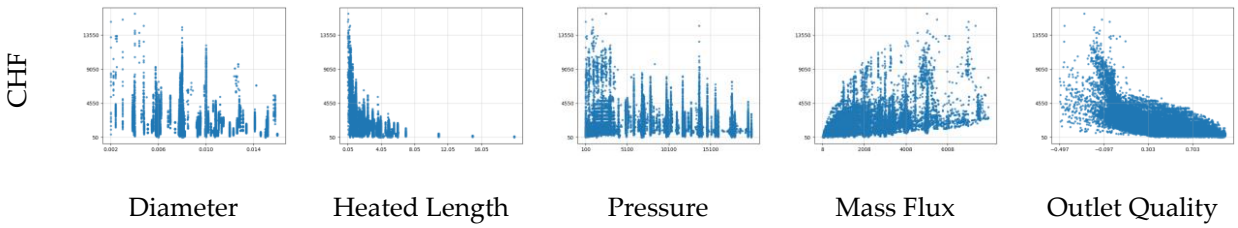


Figure 4.1: Scatter plots of CHF and inputs

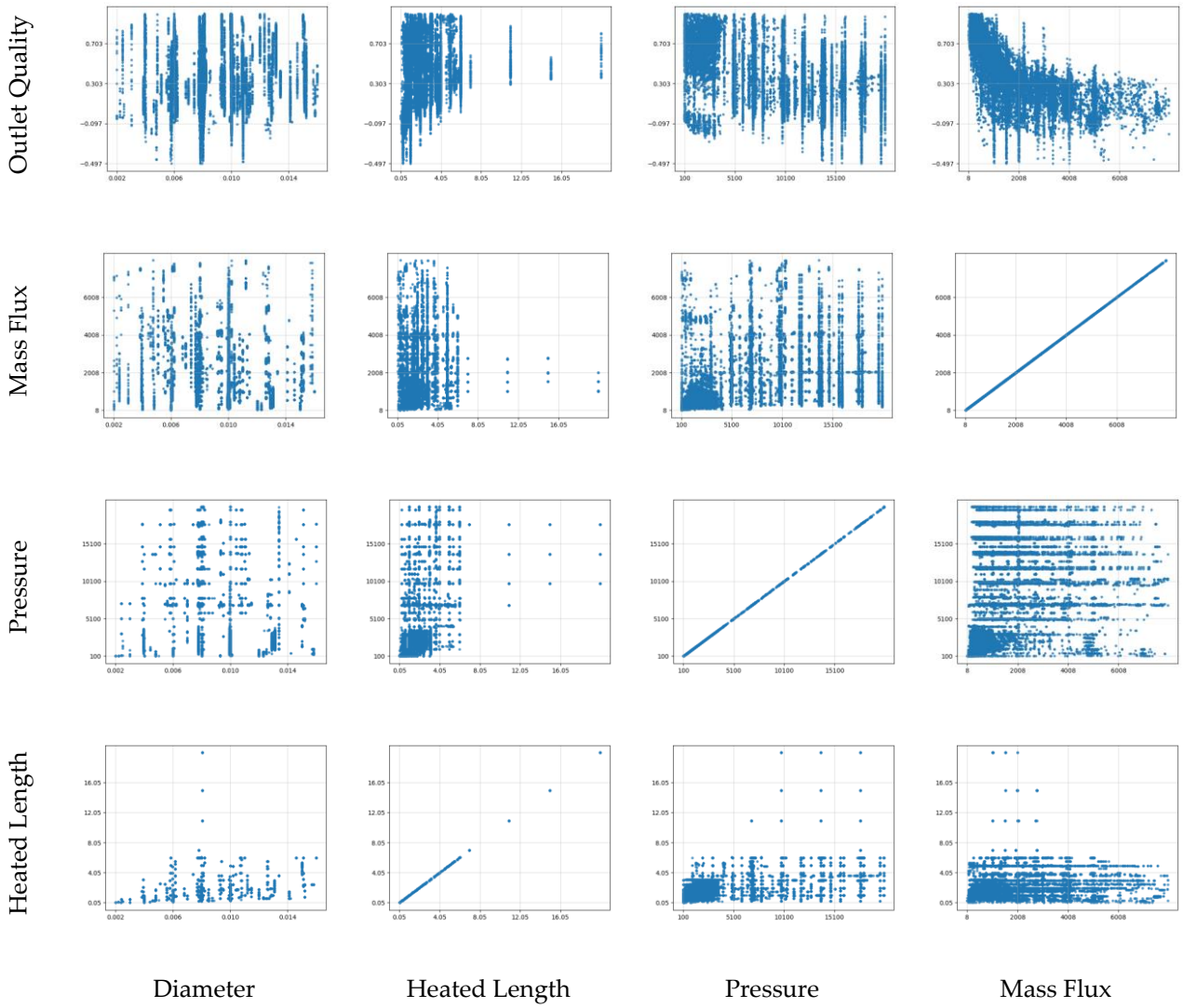


Figure 4.2: Scatter plots of input pairs of variables

4.1.1. Data Preprocessing

Proper dataset management is a crucial stage of NN development. Invalid data points, zeros, duplicates and incorrect data values can negatively affect the learning process of the NN leading to inaccurate predictions. Before starting the architectural development of the network, a thorough analysis of the type of problem and dataset used is fundamental to determine how to move forward.

Given the scales differences across all 7 available input variables, the difficulty of neural network modelling may increase [58]. Thus, data scaling, which is anyway a recommended pre-processing step for accurate machine learning algorithms, it's implemented. The training dataset is normalized using MinMaxScaler from the Python scikit-learn library, which transforms data by scaling each value in the range between 0 and 1 using the following formulation:

$$y_{scaled} = \frac{y_{original} - y_{min}}{y_{max} - y_{min}} \tag{4.1}$$

For PINN implementation, the application of MinMaxScaler presents some extra difficulties as presented in Paragraph 3.2.2. The scaling process is directly integrated into the forward and backward propagation steps. This implies transforming the tensors back into numpy arrays to apply the scaler, however, this procedure detaches the original tensor from the computational graph responsible for calculating gradients during backpropagation. Since numpy array do not support automatic differentiation, a consequent reconversion in tensors is necessary but it does not recover any previously calculated gradients. To address this issue, a custom scaler is developed, as detailed in Appendix B.1. This approach ensures that the computational graph is not broken and gradients are correctly tracked throughout the network's workflow.

4.1.2. NRC CHF Dataset vs W-3 Correlation Applicability Range

The last step for the loss function definition for PINN requires an accurate evaluation of the ranges of validity of the physical correlation with respect to the parameters ranges of the dataset used for the supervised learning. In Table 4.2 both intervals are presented for comparison, it's immediately clear that the Westinghouse-3 correlation doesn't cover the entire parameters space of the NRC dataset. For this reason, the loss function for the PINN is computed using a "for" loop that applies the total loss when the inputs fall within the Westinghouse-3 ranges of validity, and in all other cases, it computes only the data loss. In Appendix B.2 the full .py codes written for both approaches for the physics loss to define the PINN loss functions are displayed.

Table 4.2: Parameters rages W3 vs NRC dataset

Tube Diameter [m]	**Heated Length** [m]	**Pressure** [kPa]	**Mass Flux** [kg/s/m^2]	**Outlet Quality** [-]
NRC Dataset				
0.002÷0.016	0.05÷20	100÷20000	8.2÷7964	-0.497÷0.999
Westinghouse-3				
0.005÷0.018	0.25÷3.66	6.89÷15.86	1356÷6781	-0.15÷0.15

5 Results and Discussion

In this Chapter, the procedure described in detail up to now is practically implemented in PyTorch to develop ML models for the prediction of CHF. After having identified the best combinations of hyperparameters through the optimization process, the final training and validation phases of the selected best models are performed. At this point it's best practice to assess prediction performance by testing the trained neural network on unseen data to check for its generalizing capabilities. This is achieved by using the dedicated test set made of 20% of the data points from the NRC CHF dataset. This approach is applied to both standard and physics-informed neural networks. By doing this procedure, the ability of each model to extrapolate correctly the informations and predict accurately the CHF is assessed. Stands out the comparison with LUT prediction of CHF whose performance is surpassed greatly by the implementation of machine learning techniques to solve this type of problem.

In the following sections, the results concerning both individual neural network models and ensemble models are presented in Section 5.1, including also a sensitivity analysis evaluation. Finally, in Section 5.2, the outcomes related to the performance of the physics informed neural network are presented and discussed.

5.1. Results of Ensemble of Neural Networks Model

In the following sections, it will be discussed in the following order: the criteria behind the development of the NNs architectures with the similarities shared among all developed models, their optimal hyperparameters search and validation, training and testing frameworks. Finally, all results will be presented for the individual NNs, followed by a final paragraph reporting the search and definition of the optimal NN ensemble model and the comparison with the 2006 LUT and other ML approaches.

5.1.1. Design and Optimization Procedures

NN Architecture

The CHF prediction is defined as a nonlinear regression problem and as such, activation functions are included in the model development. The use of dropout as regularization method for reducing overfitting (and thus have better generalization) is

also investigated. While building the neural network, the following considerations and decisions are taken:

- *Activation function*: given the nature of the problem, the rectified linear unit (ReLU) is selected. Due to vanishing gradient problem i.e. gradient reaching the value zero, Sigmoid and Tanh activation functions are avoided (they would perform better for classification problems). ReLU is widely used for its performance and high reliability versus other activation functions in most cases [59]. LeakyReLU has also been examined but it doesn't show any significant improvement. To avoid negative values of CHF predictions on the output layer, Softplus, with optimized beta values, is used as activation function.
- *Weights initialization*: the Kaiming initialization method, specifically derived for ReLU activation functions, is selected for its proven capability of efficiently addressing the vanishing or exploding gradient problem [60] that may occur during training. It also stands out compared to other methods which are proven to underperform when compared to this approach.
- *Bias initialization*: the bias for all hidden layers is initialized to zero, as per common practice. Future need of initialization bias correction will be addressed by the optimizer during training.

For the output layer, the activation function nn.Softplus, which is a smoother approximation of the ReLU function, and nn.ReLU have both been tested. The first gradually approaches zero for negative values of the output layer, as shown in Figure 5.1, and behaves like a linear function for positive values, while the second does a sharper transition. They both ensure non-negative outputs which are unphysical for the considered problem. ReLU's sharper and simpler behavior, better captures linear and less complex relationships in the inputs, providing stronger gradients. However, it suffers from the dead neurons problem, which means that all negative neurons informations are set to zero and thus lost. LeakyReLU allows a small portion of negative neurons to move forward, yet this reintroduces negative outputs in the results. Softplus, on the other hand, offers a smoother transition and avoids the dead neurons issue, but it can be too soft to capture rapid variations in the data. To find a trade-off, Softplus is selected, but with a higher value of beta (intrinsic parameter setting the curvature). As for the number of hidden layers and nodes, as well as regularization methods such as dropout, they will be discussed in the hyperparameters search section below.

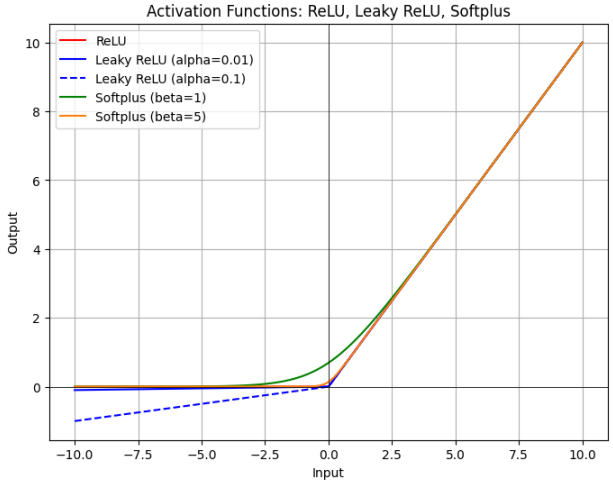


Figure 5.1: Output layer Activation Functions

Hyperparameters Search

Machine learning models are made of sets of hyperparameters that must be customized to fit the considered dataset to properly learn the intrinsic connections between inputs and output. Hyperparameters, unlike model parameters (weights and biases), are set during the validation step and are not modified during training. Selecting the optimal combination of hyperparameters (such as learning rate, batch size, number of hidden layers and nodes) and regularization method (dropout and weight decay) can drastically influence the model's performance and it generalizing ability. The learning rate controls the magnitude of the steps the model takes during gradient descent to adjourn weights and bias and thus the convergence speed. A high learning rate allows for rapid but more rough learning. On the other hand, a small learning rate ensures a more meticulous, although slower, progress. The batch size determines the size of the data subset used per each epoch before updating the internal model parameters, it balances the needs of gradient accuracy and reasonable computational time.

There are different optimization algorithms that can be used, two of the most simple and common methods are Random Search and Grid Search. They both require the definition of a search space, which is a n-dimensional volume to be searched where each dimension represents a hyperparameter and thus one possible model configuration. Random Search randomly samples points in the space search domain and tries each random hyperparameters combination that's formed. Grid Search evaluates every position in a grid made of each possible combinations of the points in the search space. This last option is ideal for checking combinations that are already known to perform well and selecting the best. Random Search usually requires more time than Grid Search, but since this work began without prior knowledge of optimal combinations of hyperparameters, Random Search is selected as the starting point.

Given the complex nature of CHF, random hyperparameters search for 1,2,3 and 4 hidden layers is performed.

An hyperparameters space is hence constructed with a pool of values for each item to pick from during each random iteration. The range for learning rate values is set between 0.1 ÷ 0.0001, batch size has options varying from 32 to 224 elements and the weight decay is tested between 0 ÷ 0.001. The number of epochs is adjusted based on early stop triggers, which was set to stop iterations after 50 cycles if no improvement is shown with respect to the best registered loss. Also, the mean squared error (MSE) is the statistical measure used for the loss function for this type of regression problem as defined in equation (3.3).

Out of the combinations tested, the best models for each number of hidden layers are further optimized by varying dropout and number of nodes. Dropout randomly sets a fraction of the nodes to zero during each forward pass to prevent the model from becoming overly reliant on any single node. Weight decay adds a penalty to the loss function based on the magnitude of the weights, discouraging excessively large weight values and promoting simpler models that are less likely to overfit. A value of zero for both dropout and weight decay has consistently proved to give the best learning curves. Increasing or decreasing those values didn't improve performance and no overfitting is observed in the selected models. The models performed better without any regularization technique likely because they are overly restrictive for this specific dataset, preventing the model from fully capturing the underlying patterns.

Table 5.1: Number of folds and subsets dimensions

train_size : 90% ; test_size : 10%				
Evaluation k value	*Validation RMSE*	*Test RMSE*	*Data within ± 10% error*	*Data within ± 20% error*
5-fold cv	0.0092	0.0060	96.91%	99.35%
10-fold cv	0.0090			
train_size : 80% ; test_size : 20%				
Evaluation k value	*Validation RMSE*	*Test RMSE*	*Data within ± 10% error*	*Data within ± 20% error*
5-fold cv	0.0098	0.0053	96.97%	99.47%
10-fold cv	0.0094			

At this point, 5-fold and 10-fold configurations for k-fold cross validation are compared. As reference, the 2 hidden layers best model is used given that it yields the best results, which will be shown in full in paragraph 5.1.4. Results are shown in Table

5.1 indicating that the 80-20 train-test split with 5-folds cross validation is the optimal choice for this analysis, given the lower computational effort required and no significant performance changes compared to the 10-folds setup. Although the 90-10 split shows slightly better validation RMSE the difference is minimal. The 80-20 split provides a better balance, as it retains more data for testing, ensuring more robust model evaluation and helps avoiding overfitting. Additionally, the 80-20 split with 5-fold cross-validation delivers a high percentage of data within ±10% and ±20% error (96.97% and 99.47%, respectively), which demonstrates the model's reliability.

Training, Validation and Testing

Having defined the NN general architecture, the training and validation phases are addressed. Validation evaluates how well the model is likely to perform in real-world scenarios. The main goal is to avoid overfitting, which is a situation where the model performs extremely well on the training data but fails to generalize to new data. During training, the model adjusts its parameters to minimize the loss function on the training dataset. However, this can lead to the model capturing noises or specific patterns that do not generalize beyond the that set. By evaluating the model on a separate validation dataset, the necessary adjustments can be made.

Before applying any validation method, the dataset is typically divided into a training set and a testing set, as previously described. There are various approaches for performing model validation, the two main ones are manually setting aside part of the training set for validation, and the cross validation technique. The first option is preferred when dealing with very large datasets, as a considerable amount of data remains available for training without making the code computationally too heavy. On the other hand, with small datasets every data point is crucial for training and removing some data for validation could compromise the model's learning process. In this work, the second option is preferred and implemented.

There are quite a few cross validation techniques available (k-fold, stratified k-fold, LOOCV, leave-p-out cross validation, etc.), the most common is the k-fold cross validation. This method separates the training dataset into k folds, as shown in Figure 5.2, then, one fold is used for validation and the others for training. This process is repeated k times, each time with a different fold used for validation, until all folds have been used for both validation and training. To monitor performance, an early stopping procedure is also common practice and it's inserted with a patience of 50 iterations as indicator to stop iterations if no improvement is shown with respect to the best registered loss.

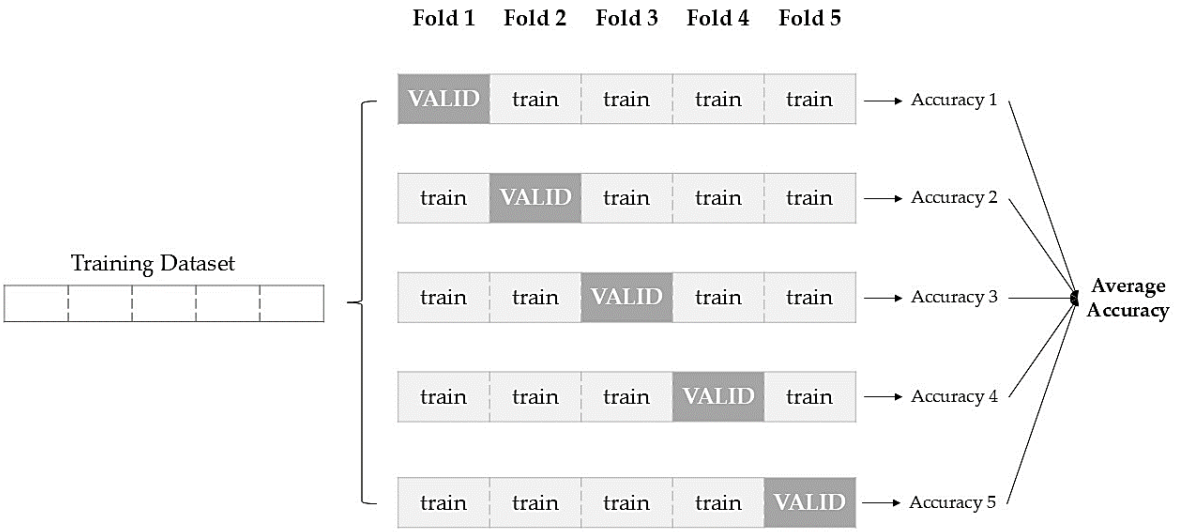


Figure 5.2: k-fold Cross Validation

Backpropagation: Optimizer

For the optimization process, the Adam [61] (Adaptive Moment Estimation) algorithm is employed for gradient-based optimization. Adam combines the advantages of two other popular optimization algorithms: AdaGrad and RMSProp. Specifically, Adam adapts the learning rate for each parameter by computing individual adaptive learning rates from estimates of first and second moments (the mean and the uncentered variance) of the gradients. This method stabilizes the training process and allows the model to converge faster and more reliably compared to traditional stochastic gradient descent (SGD).

In the context of PyTorch, after defining the model and the loss function, the backpropagation process is initiated by calling loss.backward(). This command computes the gradients of the loss with respect to each of the model's parameters (weights and biases) and stores these gradients in the .grad attribute of each parameter's tensor. These gradients represent the direction and magnitude by which each parameter should be adjusted to reduce the loss following the mathematical representation of Chapter 3.1. After computing the gradients, the next step is to update the model's parameters through the optimizer. This is achieved by calling optimizer.step(). At this point, Adam updates each parameter by applying the gradients calculated during loss.backward().

5.1.2. Results of Individual Neural Networks Models

In Table 5.2 are shown the optimal parameters, obtained from Random Search, for each of the selected individual NN models, with architectures limited to a maximum of three hidden layers. The only common factor among all those models is the fact that no regularization technique (dropout and weight decay) is needed for better model

performance. The architecture made of four or more hidden layers are excluded due to increased error and a noisy learning curves. In total, seven optimized NN models are developed (M = 7), as summarized in Table 5.2, with their respective cross validation and test performances reported in Table 5.3. Among these models, Model D achieved the lowest cross validation RMSE during training, slightly surpassing Model A. However, considering test set performance, Model A consistently outperforms all other models, including the LUT, across nearly all metrics, except for the Q^2-error, where Model B excels. Overall, Model A is identified as the strongest individual model.

These results underscore that even when models share some architectural configurations, variations in training parameters, such as learning rate, can lead to different performances on new, unseen data. Notice that Models A and B, with identical architecture but differing in learning rate and number of epochs, perform differently on the test set, with Model A significantly outperforming Model B across all metrics, apart from Q^2-error. Additionally, increasing the number of layers does not necessarily lead to improved performance; for instance, while Models D and E (three hidden layers) exceed the performance of Models B and B2 (two hidden layers), they underperform relative to Model A, which also utilizes two hidden layers. These findings indicate that each individual model has distinct strengths and limitations, suggesting that an ensemble model combining the optimized individual models could enhance their individual strengths while minimizing weaknesses.

Table 5.2: Optimal parameters of the obtained individual single models

ID	**1 Hidden Layer**	**2 Hidden Layers**			**3 Hidden Layers**		
	F	*A*	*B*	*B2*	*C*	*D*	*E*
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Loss Function	MSE	MSE	MSE	MSE	MSE	MSE	MSE
Nodes	7/120/1	7/70/60/1	7/60/60/1	7/60/30/1	7/60/40/20/1	7/70/50/40/1	7/70/60/50/1
Beta	2	4	4	4	1	5	4
Dropout	0	0	0	0	0	0	0
Learning rate	0.01	0.001	0.0001	0.01	0.01	0.0001	0.001
Epochs	150	250	1400	200	200	650	200
Batch size	164	24	24	112	32	24	24
Weight decay	0	0	0	0	0	0	0

Table 5.3: CV and test results of all the obtained optimal individual models

ID	Cross Validation	Training	Testing				
	RMSE	*MSE*	*RMSPE*	*MAPE*	Q^2	*data within ± 10% error*	*data within ± 20% error*
LUT	-	-	43.05	22.31	0.0606	43.10%	67.05%
Model F	0.0140	7.25E-05	15.60	5.53	0.0048	89.14%	96.58%
Model A	0.0098	3.23E-05	6.53	2.91	0.0031	96.97%	99.47%
Model B2	0.0116	5.72E-05	11.03	4.45	0.0039	92.49%	97.52%
Model B	0.0104	2.22E-05	7.95	3.31	0.0023	95.28%	98.88%
Model C	0.0111	7.57E-05	12.75	4.87	0.0050	91.72%	97.88%
Model D	0.0097	2.21E-05	8.12	3.24	0.0027	96.16%	98.88%
Model E	0.0103	3.65E-05	6.57	3.48	0.0037	95.30%	99.02%

Figure 5.3 shows a comparison between the best single model (Model A) and LUT performances, comparing predicted versus measured CHF results particularly referring to a 10% error margin, which shows the most significant difference in accuracy. In general, considerable improvement is shown with respect to LUT predictions among all models evaluated. Particularly, models A and D consistently show some of the best metrics results, indicating good learning of input-output relationship and generalizing capabilities of the NNs.

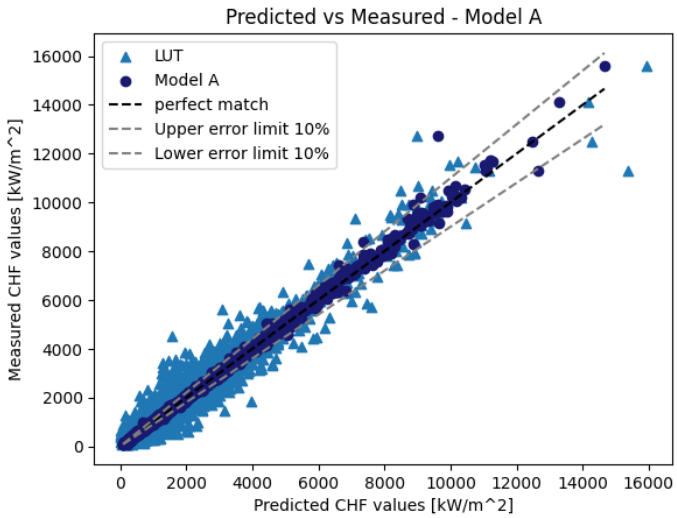


Figure 5.3: LUT vs NN model A prediction accuracy

5.1.3. Results of Ensemble of Neural Networks Models

Based on the analysis of the developed individual models presented in Section 5.1.2, models with suboptimal performance are excluded from consideration in the ensemble model optimization. Specifically, five models (Models A, B, C, D and E) are selected for their robust performance across all metrics (Table 5.3). Models F and B2 are excluded due to weak cross validation and poor testing outcomes (Table 5.3). Given the five selected models ($M = 5$), 26 possible ensemble configurations (i.e., $N^e = 26$) are identified, comprising 10 configurations with two models, 10 configurations with three models, 5 configurations with four models, and 1 configuration with all five models. Each ensemble configuration is subjected to the same cross validation and evaluation metrics used previously on the 80% training dataset. Specifically, both RMSPE and MAPE are calculated for each configuration and two aggregation methods (mean and median), as described in Section 3.1.2, are tested, with only the results of the best-performing aggregation method being here presented.

Figure 5.4, Figure 5.5 and Figure 5.6 show the performance of all 26 ensemble configurations. Figure 5.4 presents the results from the 10 ensemble configurations combining two individual models. Here, both mean and median aggregation methods yield identical outcomes, as each ensemble consists of only two models. Based on these results, the configuration $\{A, E\}$ demonstrates the best RMSPE (6.98), whereas the configuration $\{A, B\}$ (with RMSPE of 7.40) achieves the best MAPE (2.33), slightly better than that of the ensemble $\{A, E\}$ (MAPE of 2.67). When comparing these results to the 10 configurations of three models (Figure 5.5), most ensembles of three models outperform those with two. Similarly, the performance of most four-model ensembles (Figure 5.6a) surpass that of the two- and three-models configurations. Overall, as the number of individual models in an ensemble increases, performance generally improves. However, despite these improvements, many ensembles with fewer models substantially outperform some that include a greater number of models. For example, the ensemble configuration $\{A, B, E\}$ significantly outperforms the configuration $\{A, B, C, D, E\}$ across both metrics. This finding confirms the importance of optimizing ensemble models to identify the most effective configuration rather than arbitrarily combining multiple models into an ensemble.

Ensemble Model $\{A, D, E\}$ achieves the best RMSPE (6.82) with the median aggregation method, whereas Ensemble Model $\{A, B, D, E\}$ demonstrates superior performance in terms of MAPE (2.06) using the mean aggregation method. Since the RMSPE metric is sensitive to outliers, where larger deviations from actual values have a greater impact on calculations, Ensemble Model $\{A, B, D, E\}$ is selected as the final optimal configuration. The proposed optimization procedure effectively identified this configuration as optimal, illustrating the benefit of systematic selection over arbitrary model combinations.

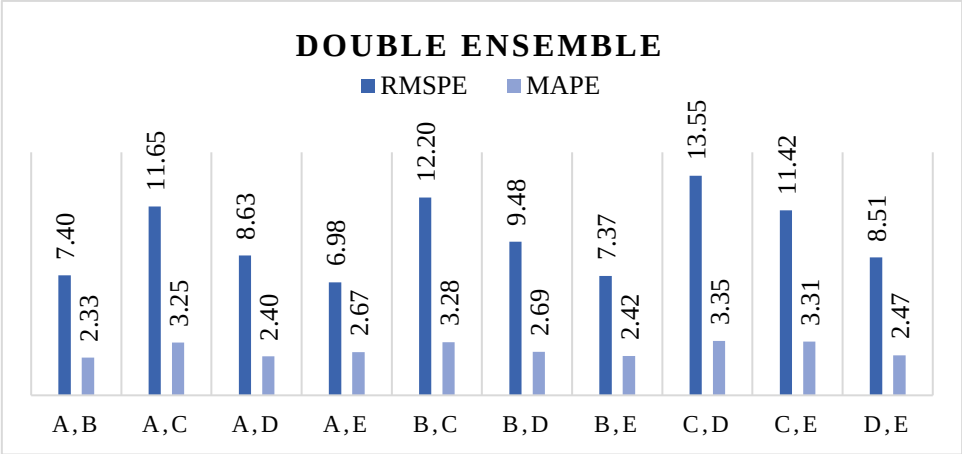


Figure 5.4: Performance of the ensembles configurations of two individual models on training set CV

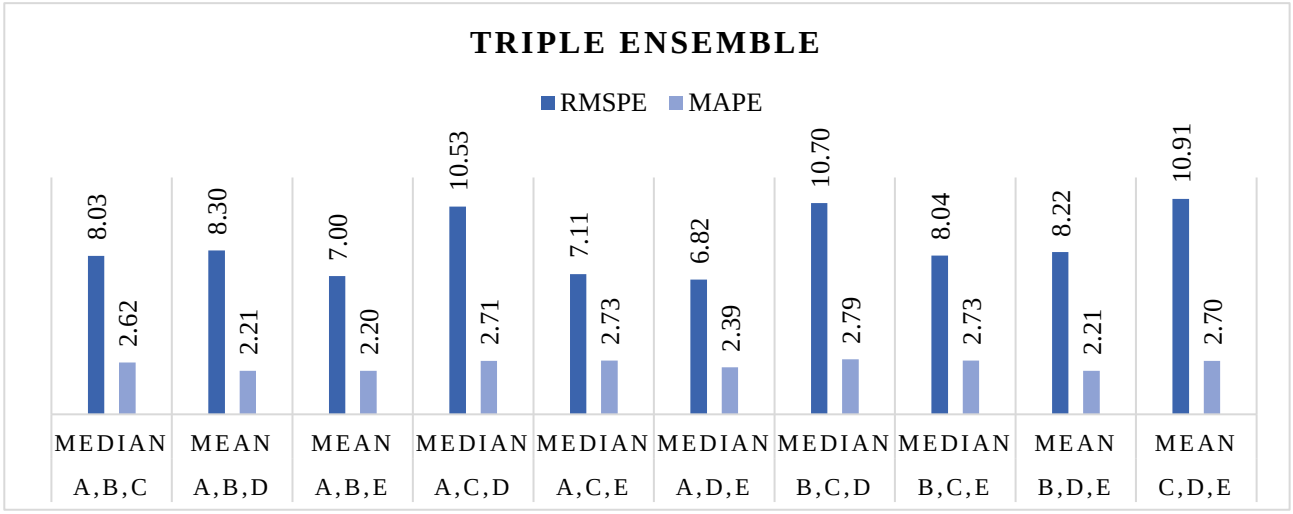
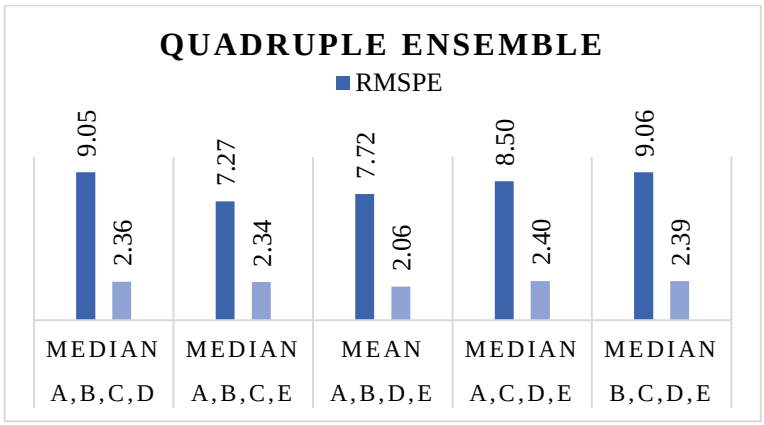
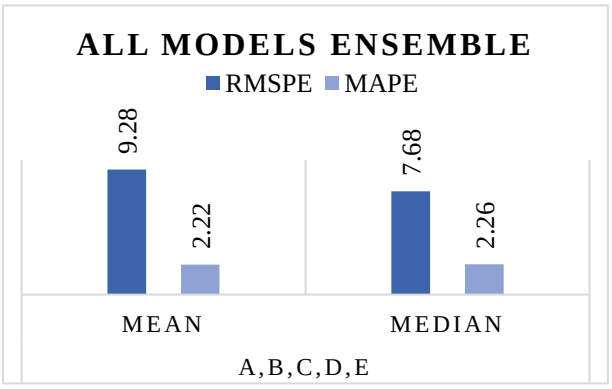


Figure 5.5: Performance of the ensembles configurations of three individual models on training set CV



(a) Four Models Ensemble



(b) Five Models Ensemble

Figure 5.6: Performance of ensemble of: (a) four and (b) all five individual models on training set CV

Figure 5.7 compares the CHF prediction of the proposed ensemble model, the best-performing individual model (Model A) and the LUT method on the test dataset with respect to the measured experimental values. The results indicate that a higher proportion of CHF data points from the ensemble model fall within a ±10% error margin than those from both the best individual model and the LUT. This indicates the effectiveness of the ensemble model in capturing input-output relationships and its robust generalization capabilities. Figure 5.8 presents scatter plots of the predicted-to-measured CHF ratio (P/M) against various input parameters for both the proposed ensemble model and LUT. An ideal P/M ratio close to 1 across the test set indicates desirable predictive accuracy; P/M ratios greater than 1 suggest overestimation, whereas ratio values lower than 1 indicate underestimation. As shown in Figure 5.8, the proposed ensemble model significantly enhances CHF prediction accuracy across multiple key parameters, including tube diameter, heated length, inlet subcooling, inlet temperature, mass flux, pressure and outlet quality. These results demonstrate the improved generalization capabilities of the proposed ensemble method, highlighting its potential for broader applications within the thermal engineering field.

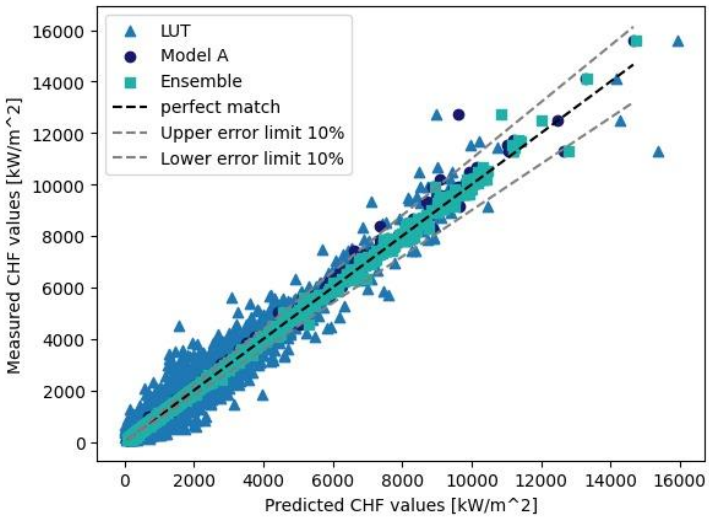
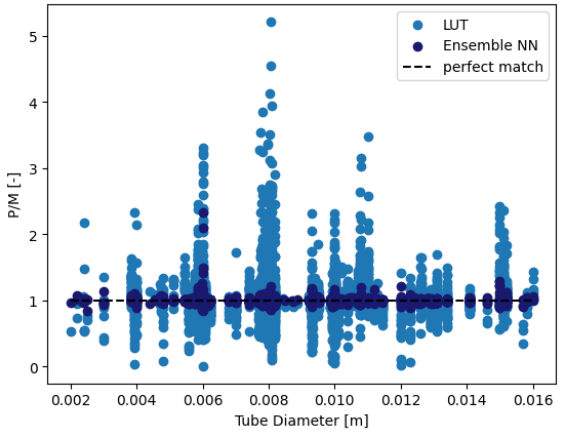
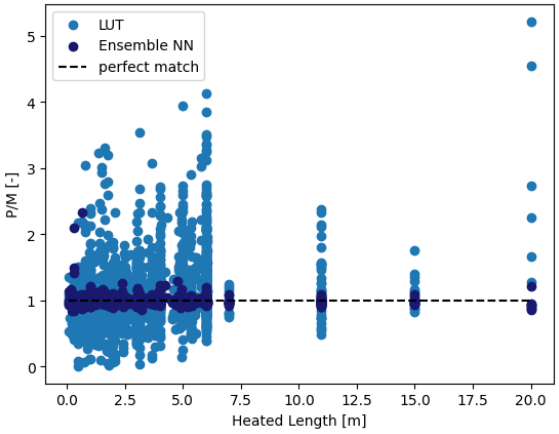
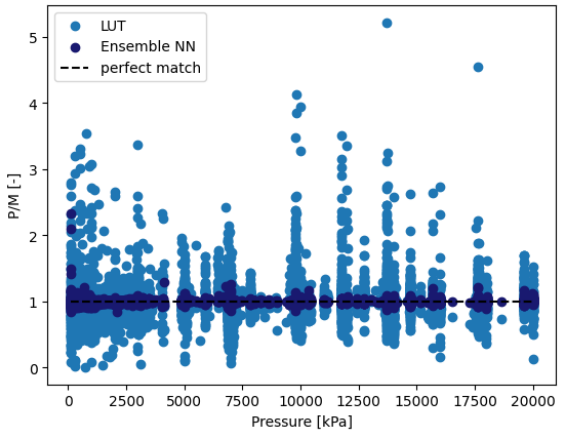
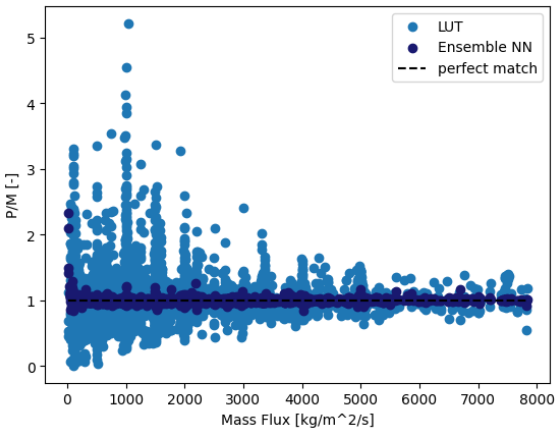


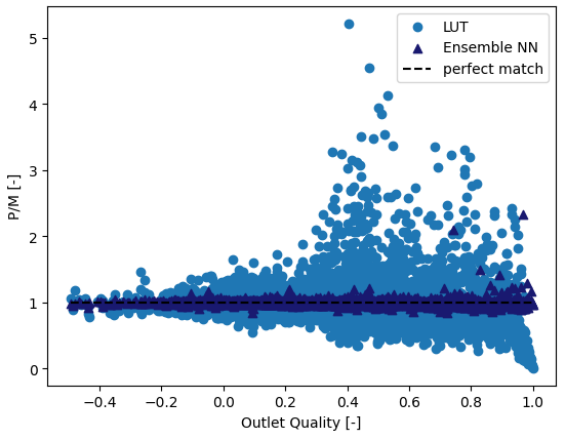
Figure 5.7: Measured vs. predicted CHF on test data for the optimal ensemble model, best single model and LUT

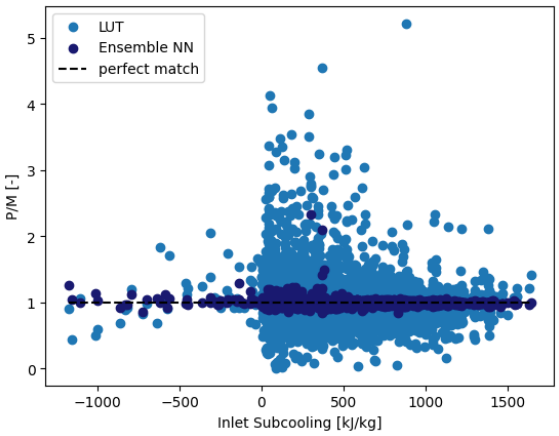

LUT
Ensemble NN
perfect match
P/M [-]
Tube Diameter [m]


LUT
Ensemble NN
perfect match
P/M [-]
Heated Length [m]


LUT
Ensemble NN
perfect match
P/M [-]
Pressure [kPa]


LUT
Ensemble NN
perfect match
P/M [-]
Mass Flux [kg/m^2/s]


LUT
Ensemble NN
perfect match
P/M [-]
Outlet Quality [-]


LUT
Ensemble NN
perfect match
P/M [-]
Inlet Subcooling [kJ/kg]

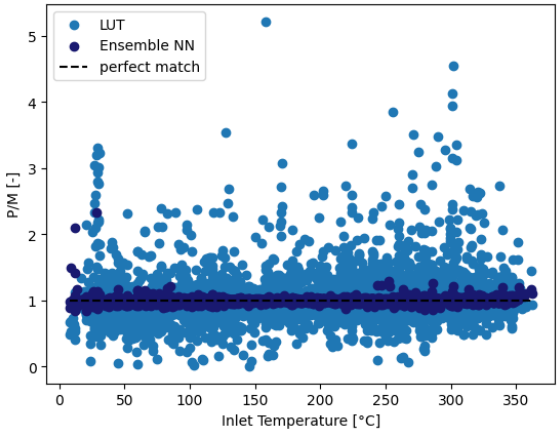


Figure 5.8: Scatter plots of LUT vs NN Ensemble for each input parameter

The performance of the proposed ensemble method has been verified by its comparison to those of three state-of-the-art prediction methods based on LUT, Random Forest (RF) and Support Vector Regression (SVR), respectively. Both RF and SVR are trained and optimized using the same 80% training set used to develop the proposed method. Table 5.4 reports the values obtained by the different methods for the performance metrics. The performance metrics indicate that all the NN models (both best individual and ensemble models) significantly outperform all other methods, with the ensemble model showing superior performance across all metrics: the ensemble model not only achieves the lowest RMSPE and MAPE values but also indicates the smallest value of Q^2-error (nearly 0), indicating strong predictive capability of the ensemble model. Moreover, the ensemble model significantly improves the prediction accuracy with 98.35% and 99.55% of the predicted data points falling within the 10% and 20 % error bands, respectively, values which are larger than those of the other models.

Table 5.4: Comparison of the accuracy of CHF prediction models

ID	**Testing Dataset 20%**				
	RMSPE	*MAPE*	Q^2	*data within ± 10% error*	*data within ± 20% error*
Look Up Table	43.05	22.31	0.0606	43.10%	67.05%
Individual NN Model - A	6.53	2.91	0.0031	96.97%	99.47%
Ensemble NN Model - A,B,D,E	5.94	2.04	0.0017	98.35%	99.55%
Random Forest Regression (RF)	7.97	4.64	0.0115	89.20%	97.84%
Support Vector Regression (SVR)	15.53	8.03	0.0063	77.36%	91.78%

The results demonstrate that the ensemble model significantly outperforms the traditional look-up table method across all metrics, and it also shows a slight improvement with respect to individual models. Also, more generally, machine learning techniques, of any kind, have proven to better capture the relationships between CHF-related parameters for improved accuracy in its prediction with respect to the LUT (the 2006 Groeneveld LUT) method. To further the assessment on the optimal set of models for the ensemble NN, a sensitivity analysis is presented in the following paragraph to evaluate the behavior of the models presented (ensemble and individual) with varying input parameters.

5.1.4. Sensitivity Analysis

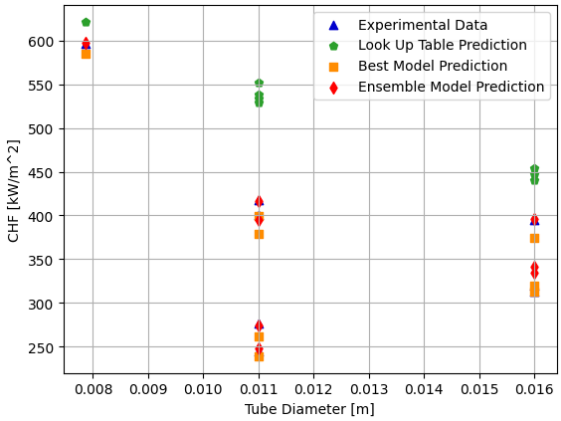
To further validate the proposed method, a sensitivity analysis is performed using the slicing approach from the NRC CHF dataset as described in [16]. Regions with varying values of diameter, heated length, pressure, mass flux and outlet quality have been identified, while the other parameters have been held approximately constant. From these regions, two slices for each varying parameter have been selected, as shown in Table 5.5. The parameters in gray are the varying ones, while the others are approximately constant; a small interval is taken in some cases to extract a higher number of rows. These "sliced" datasets allow for the demonstration of the physical behavior of the CHF prediction model across the parameter space, enhancing confidence in the predictive capabilities of the model and in preventing overfitting.

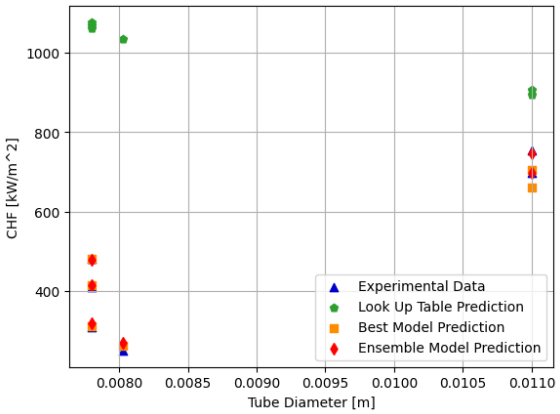
Figure 5.9 shows the results of the predictions for all Slices for the ensemble model, the best-performing individual model and the LUT. To assess the prediction accuracy of the proposed method, performance metrics are calculated. Table 5.6 reports these metrics, comparing the ensemble model with the LUT and the best-performing individual model, which showed impressive performance in Section 5.1.3. The ensemble model consistently outperforms other methods across nearly all sliced datasets and all metrics (except for Slice 05 for which the best individual model achieves the lowest MAPE value). These results highlight the robust predictive capabilities of the ensemble model and its consistency with expected physical behaviors.

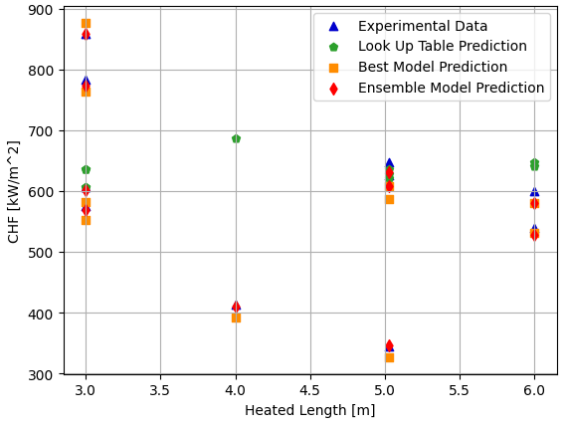
Table 5.5: Ranges of slice datasets

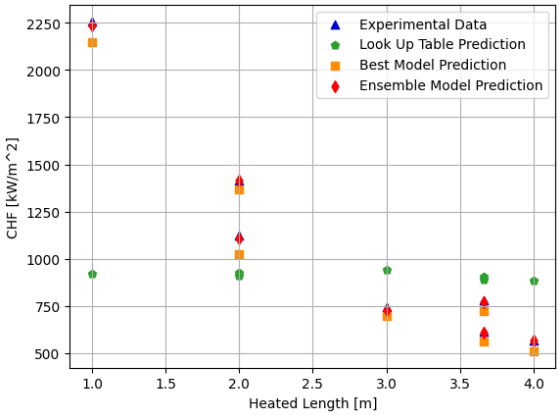
	D	**L**	**P**	**G**	**X**	*# points extracted*
	[m]	*[m]*	*[kPa]*	*[kg/m^2/s]*	*[-]*	
S01_extr	0 – 0.016	6	14700 – 14710	988 – 1009	0.388 – 0.394	8
S02_extr	0 – 0.016	6	9800 – 9810	999 – 1009	0.527 – 0.532	6

S03_extr	0.008 – 0.00803	0 – 20	9800 – 9810	993 – 1002	0.582 – 0.59	10
S04_extr	0.00811	0 – 20	2000 – 2020	749.9 – 755.9	0.751 – 0.762	7
S05_extr	0.00799 – 0.00801	0.99 – 1	0 – 20000	1993 – 2014	0.138 – 0.142	9
S06_extr	0.0134	3.658	0 – 20000	2022.7 – 2055.1	0.369 – 0.387	35
S07_extr	0.008	1.57	12750	0 – 8000	0.142 – 0.147	11
S08_extr	0.01	4.966	16000	0 – 8000	0.339 – 0.348	9
S09_extr	0.00795 – 0.0085	1.8 – 2	9800 – 10050	1250.9 – 1672.2	-0.5 – 1	28
S10_extr	0.008 – 0.00801	0.99 – 1	17650	1977 – 2014	-0.5 – 1	54









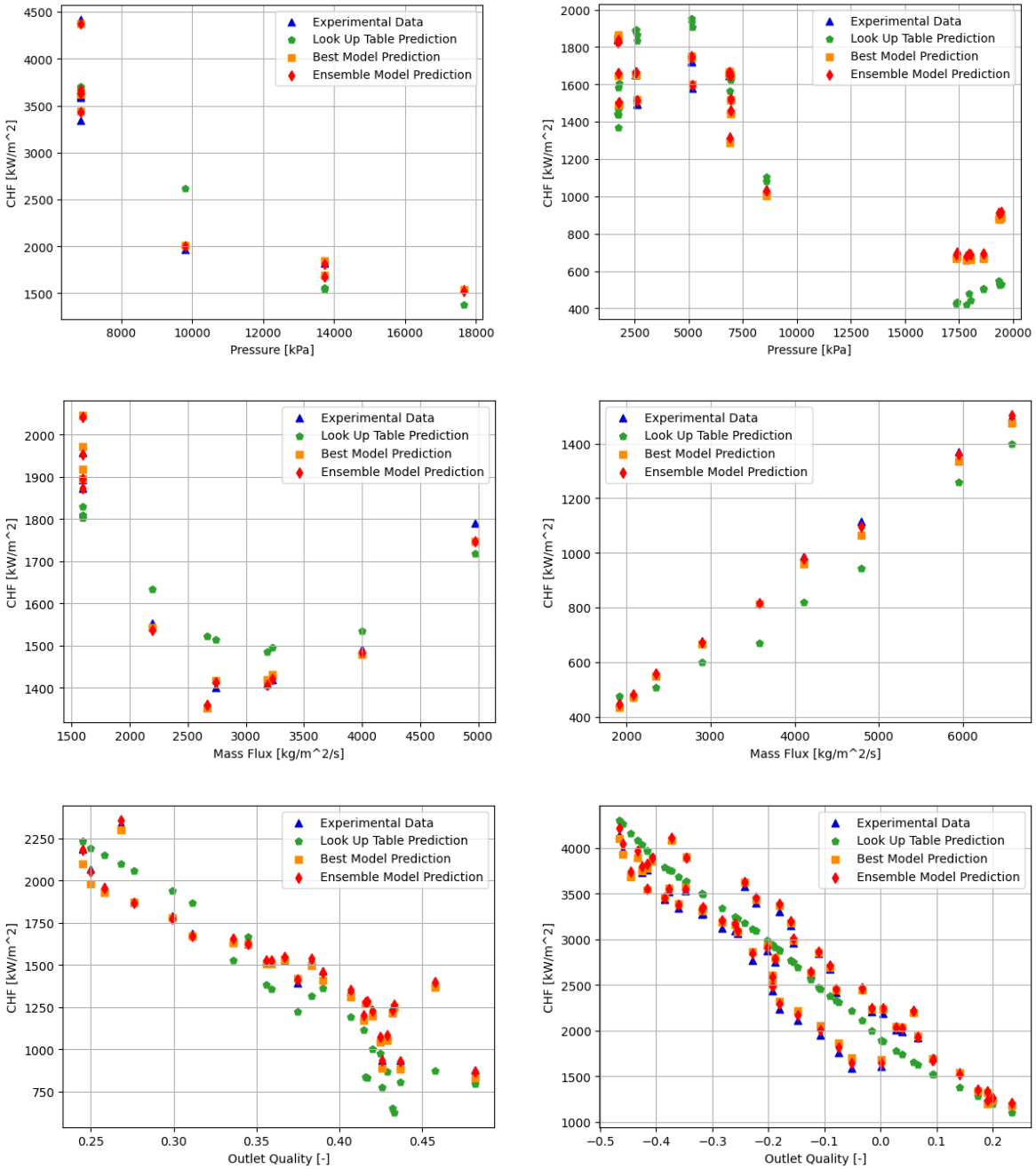


Figure 5.9: Slices plots LUT vs Individual vs Ensemble

Table 5.6: Comparison of the accuracy of CHF prediction models for slice datasets

Extracted Data		S01	S02	S03	S04	S05	S06	S07	S08	S09	S10
RMSPE	*LUT*	60.55	182.65	35.37	40.10	14.51	22.59	6.97	11.55	20.05	13.71
	Individual	3.78	4.06	4.32	7.37	1.42	1.95	1.10	2.79	2.61	2.37
	Ensemble	3.41	3.61	1.89	1.29	1.39	1.19	0.88	0.97	0.63	1.73
MAPE	*LUT*	47.18	148.33	23.41	36.49	11.15	18.64	6.35	10.35	15.35	11.26
	Individual	3.11	3.03	4.04	7.03	1.13	1.53	0.90	2.48	2.04	1.79
	Ensemble	2.19	2.41	1.58	1.03	1.19	0.95	0.57	0.80	0.54	1.53
Q^2	*LUT*	8.136	58.354	51.227	936.160	0.132	0.203	0.701	0.116	0.266	0.116
	Individual	0.020	0.025	0.028	0.018	0.002	0.002	0.006	0.005	0.010	0.004
	Ensemble	0.012	0.003	0.007	0.001	0.002	0.001	0.004	0.001	0.001	0.003

5.2. Results of the Physics-Informed Neural Network Model

The implementation of the PINN framework has been discussed in Chapter 3.3, with specific focus on defining the loss function. Before integrating the physics into the model, it is necessary to optimize the baseline neural network, in order to establish a solid foundation upon which the PINN is to be built. An over-parametrized model could lose its ability to effectively generalize by becoming overly complex and by capturing noises in the training data as actual inputs' patterns. To mitigate this potential problem, the number of input variables is reduced from seven to five. The results are then compared with the previous ones to study the effect of reducing the number of inputs. Inlet temperature and inlet subcooling are thus excluded because they are usually considered less critical in influencing the CHF compared to the other parameters.

After this adjustment, the neural network is optimized using, as for the standard NNs, the MSE as loss function. Once this baseline model is established, the next step is to integrate the physics-based loss into the total loss function as defined in equation (3.11). To do this, the lambda parameter, which weights the relative importance of each component of the total loss, is determined through validation to ensue optimal learning behavior. Once this last parameter is defined, testing results are collected for both considered approaches for calculating the physical loss. The overall advantages

and disadvantages of this ML method are hence examined, presenting general considerations for the use of PINN to accurately predict CHF.

5.2.1. 5 Inputs Optimization

Given the reduced number of input nodes, and thus the fewer available informations for the network learning phase, it's expected a higher number of hidden layers and related nodes to be necessary. In fact, the same architectures used for the 7-inputs NNs, after testing, do not show the same promising results. The number of hidden layers and nodes is thus increased and the models are optimized through the same Random Search procedure previously applied. The results from this search are summarized in Table 5.7.

Table 5.7: 5 inputs optimization results

ID	3 Hidden Layers	4 Hidden Layers	5 Hidden Layers	6 Hidden Layers	7 Hidden Layers	8 Hidden Layers
	Model 3	*Model 4*	*Model 5*	*Model 6*	*Model 7*	*Model 8*
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam
Loss Function	MSE	MSE	MSE	MSE	MSE	MSE
Nodes	90/70/70	80/70/70/60	90/70/70/60/60	90/80/70/70/60/50	80/70/60/60/50/40/30	80/80/70/60/50/40/30/20
Beta	5	3	1	1	10	3
Dropout	0	0	0	0	0	0
Learning rate	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
Epochs	1400	550	900	600	600	550
Batch size	24	24	24	16	64	64
Weight decay	0	0	0	0	0	0

It's immediately observed that, as for the 7-input models, regularization techniques are not necessary to manage overfitting. Moreover, the learning rate appears to be optimal at the fixed value of 0.0001 across all hidden layers configurations investigated; unlike the models with seven inputs, whose learning rate varies depending on the hyperparameters and architecture's settings. The number of epochs is related to how many iterations are necessary to have 50 consecutive epochs without any error improvement (early stopping procedure). Having defined the best sets of hyperparameters and NN architectures, the results from validation, training and testing of such models are collected and presented in Table 5.8. The same CHF dataset

is used, with the same 80-20 train-test split. From the results, the following conclusions can be made:

- The results are still better than LUT's, however not as good as those from the seven inputs models. Specifically, the results from testing suggest that limiting the number of input parameters does not improve the generalizing ability of the NN, but it's reduced due to the lower number of informations provided. This result can be expected given that the number of parameters available for the studied CHF problem is small from the beginning.
- Model 5 demonstrates the best performance across all metrics in each phase, making it the most accurate and reliable representing PINN. In Figure 5.10, the predicted CHF values using Model 5 and the measured ones are plotted for visual accuracy comparison.

Table 5.8: 5-Inputs single model results vs LUT prediction

ID	**Cross Validation**	**Training**	**Testing**				
	RMSE	*MSE*	*RMSPE*	*MAPE*	Q^2	*data within ± 10% error*	*data within ± 20% error*
Look Up Table	-	-	43.05	22.31	0.0606	43.10%	67.05%
Model 3	0.01578	1.11E-04	16.88	10.49	0.0196	63.95%	87.31%
Model 4	0.01594	1.26E-04	17.18	10.55	0.0217	64.12%	86.86%
Model 5	0.01564	9.38E-05	16.61	10.13	0.0175	66.62%	88.20%
Model 6	0.01599	1.08E-04	17.06	10.41	0.0218	65.66%	86.98%
Model 7	0.01622	1.18E-04	18.58	11.29	0.0214	62.96%	85.11%
Model 8	0.01585	1.21E-04	17.18	10.42	0.0188	65.42%	87.96%

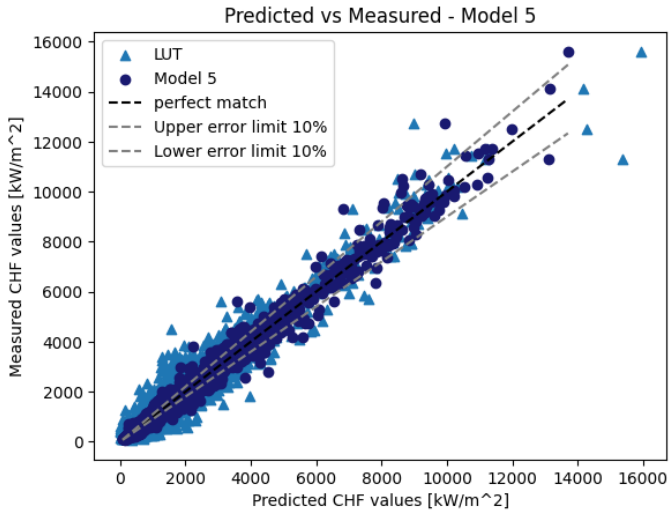


Figure 5.10: Best 5-inputs model predicted vs measured plot

5.2.2. Lambda Optimization

In order to implement the PINN's loss function as defined in equation (3.11) for both proposed methods, the hyperparameter lambda needs to be optimized to find the best value for the weight which regulates the importance of the two loss factors. In this context, the results from k-fold cross validation using model 5, which has proven to be the best one from hyperparameters optimization, are gathered. Both approaches for the physics loss are carried out and the average mean absolute error (MAE) among all k-folds for each lambda value are presented in Table 5.9.

Table 5.9: Optimal lambda values

$L_{physics}$ residual: Simple Difference	
Lambda	*Cross Validation Avg. MAE*
0	0.00969
0.1	0.01005
0.2	0.01003
0.3	0.01017
0.4	0.01017
0.5	0.01066
0.6	0.01066
0.7	0.01078

$L_{physics}$ residual: Partial Derivative	
Lambda	*Cross Validation Avg. MAE*
0	0.00969
0.1	0.01672
0.2	0.01817
0.3	0.01868
0.4	0.01937
0.5	0.01979
0.6	0.02056
0.7	0.02097

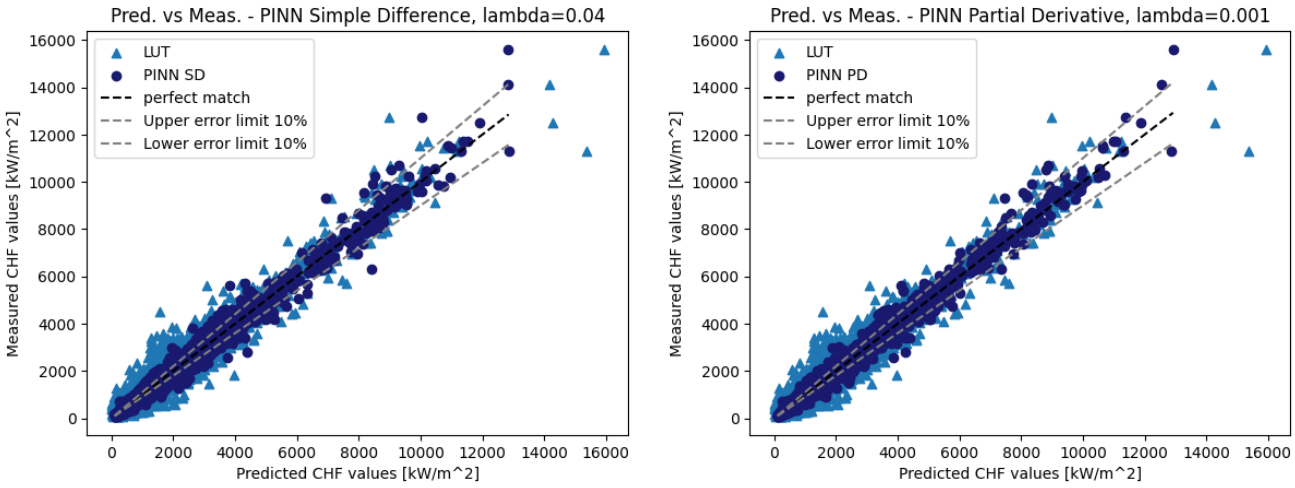
0.8	0.01127	0.8	0.02105
0.9	0.01172	0.9	0.02147
1	0.01195	1	0.02168
0.02	0.00971	0.001	0.01045
0.038	0.00968	0.01	0.01294
0.04	0.00962	0.02	0.01424
0.042	0.00969	0.04	0.01535
0.06	0.00986	0.06	0.01625
0.08	0.00997	0.08	0.01660

The MAE between actual and predicted CHF shows no significant improvement compared to the baseline model, which uses only a data-driven loss function (i.e. lambda value of zero). In the initial analysis, conducted with lambda values ranging from 0 to 1, incorporating the physical loss term into the total loss does not show any performance improvement. The best results are obtained with lambda equal to 0.2 for the simple difference method and 0.1 for the partial derivative method. A consequent refinement of the lambda value within the 0 to 0.1 range revealed that the simple difference method achieved slightly better performance at a lambda of 0.04 compared to using only the data-driven loss. However, this marginal improvement should be considered as an isolated case and it's not worth implementing given the negligible enhancement and the substantially higher computation cost associated with PINN. Also, the overall trend indicates worse performance in all other cases, and such a low value of lambda indicates that only 4% of the importance of the total loss function is due to the physical part, which is quite negligible. For the partial derivative approach, no improvements are observed even within this smaller interval. The lambda values highlighted in gray in Table 5.9 are now passed through the training and testing sections of the code and the results are presented in Table 5.10.

Table 5.10: PINN models training and testing results

ID	**Training**	**Testing**				
	MSE	*RMSPE*	*MAPE*	Q^2	*data within ± 10% error*	*data within ± 20% error*
Look Up Table	-	43.05	22.31	0.0606	43.10%	67.05%
PINN.SD – 0.04	1.58E-04	17.00	10.09	0.0171	67.39%	88.16%
PINN.SD – 0.2	1.75E-04	17.06	10.31	0.0180	66.11%	87.61%

PINN.PD – 0.001	1.91E-04	16.38	10.29	0.0168	64.97%	87.08%
PINN.PD – 0.1	2.01E-03	35.75	26.57	0.1115	24.25%	47.70%



(c) $L_{physics}$ with Simple Difference

(b) $L_{physics}$ with Partial Derivative

Figure 5.11: Predicted vs measured PINN plots

In Figure 5.11, the predicted vs measured plots are presented to display the accuracy of the models developed with the PINN framework. For the simple difference method it's considered a lambda value of 0.04 and for the partial derivative a lambda value of 0.001, as these are the two configurations that showed the best performance considering a lambda value different from zero.

One possible explanation for the presented results could be attributed to the high inaccuracy level of the CHF correlation, which may be steering the physics-informed neural network toward a less accurate learning path than the one it would follow with a purely data-driven learning. Unfortunately, the high level of uncertainty in predicting CHF using the currently available correlations is a well-known issue among all of them. Therefore, a different result is not expected with empirical correlations similar to the one used here. However, it would be interesting to test more modern correlations, which make use of empirical parameters derived from a larger dataset than the one available for the Westinghouse correlation. With access to more extensive datasets it would be valuable to observe how these updated correlations perform.

6 Input Analysis

In this final chapter, an in-depth analysis of the input variables is presented to better understand their influence on the model's predictions and obtain some insights on the PINN results. Also, this will help to assess the relative importance and impact of each feature on the output prediction, which is crucial for optimizing model performance and model's interpretability. As previously mentioned, neural networks are often referred to as 'black box' models due to their lack of a transparent decision-making process. This vague NN characteristic leads to the necessity to build a level of trust in the model's predictions before employing it [62]. To address this, interpretability instruments are leveraged.

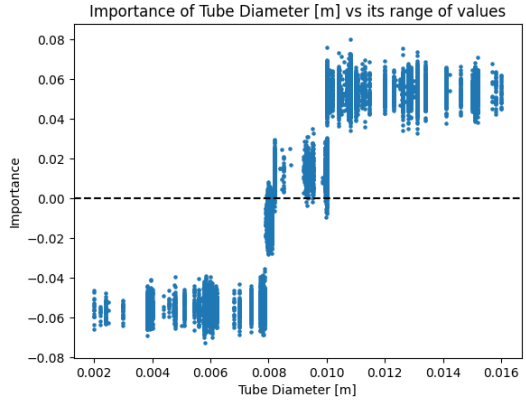
The following analysis is divided into two sections. In the first section, LIME (Local Interpretable Model-Agnostic Explanations) is used to investigate how individual features affect local predictions. The second section focuses on SHAP (SHapley Additive exPlanations), providing a more global perspective on feature importance. Both previous approaches focus on understanding how the developed machine learning models determine the output starting from the provided inputs.

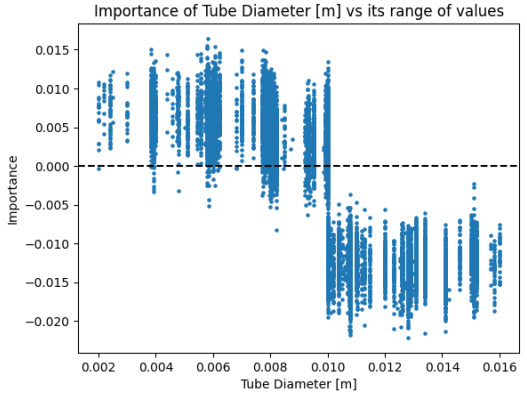
6.1. LIME

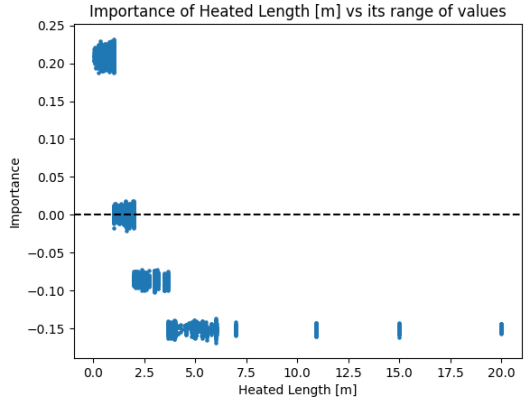
LIME (Local Interpretable Model-agnostic Explanations) provides insights into how different features contribute to individual predictions. As a model-agnostic technique, LIME can be applied to any classification or regression algorithm, allowing users to interpret models behavior on a local scale without altering their structure. By approximating the model locally, around a specific instance (a single data point, or a single row of inputs from a dataset) LIME generates explanations that show the contribution of each feature to the specific prediction.

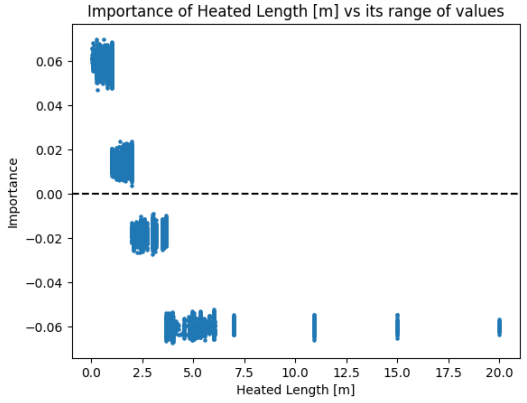
LIME achieves this by creating perturbed versions of the input data, making slight modifications to the original values, and observing the changes in the model's predictions. A simpler, interpretable model (such as a linear regression or decision tree) is then used to approximate the complex model's behavior in the immediate vicinity of the selected instance. The goal of this local interpretability approach is to understand "why" a specific prediction is made. In the context of this specific work, LIME is applied to the PINN Model 5, with the simple difference-based physics loss function, with parameter lambda equal to 0.04 for the total loss function, and to the Individual NN Model A. The results from this application are shown in Figure 6.1.

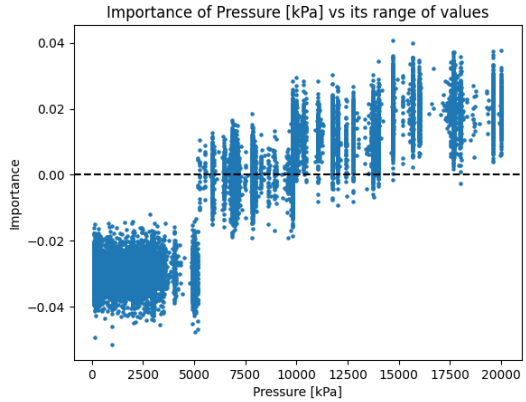
Both results are obtained considering all rows of the 80% dataset used for training and validation and they also provide an insight into the outcomes of the model developed considering 7 inputs (Model A) and the one developed considering 5 inputs (Model 5).

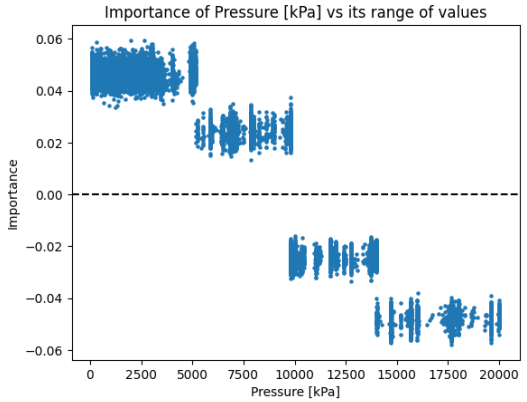












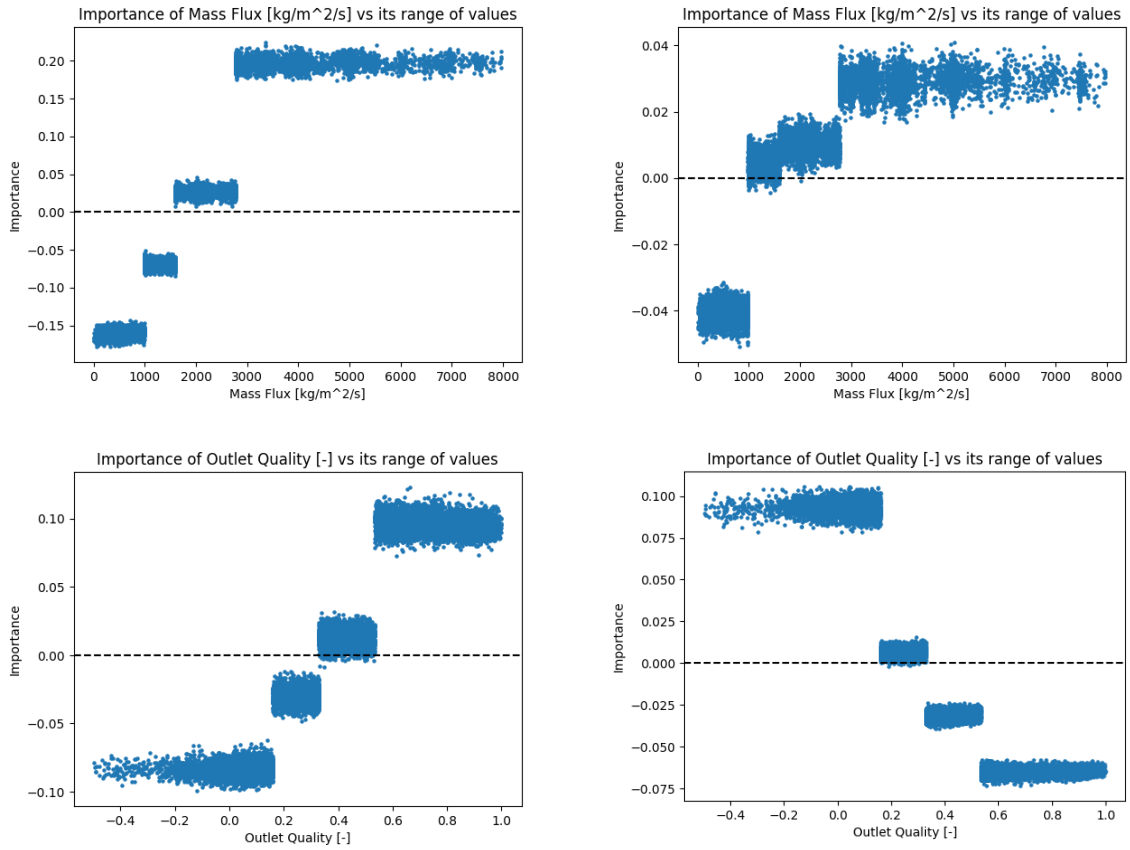


Figure 6.1: LIME plots for Individual NN Model (left) and PINN Model (right)

From the presented results, a different qualitative and quantitative behavior between the models representing a standard NN and the PINN is observed. Analyzing the qualitative behavior, the features tube diameter, pressure and outlet quality present an opposite trends for NN and PINN. The behavior shown by PINN aligns to the one from which the W-3 correlation has been empirically derived, as evidenced in [1]; this proves that the physics principles inserted in the NN through the physics loss function have been successfully integrated in the PINN model. The following considerations are derived from the three plots of the differing features:

- **Tube diameter**: experimental studies show that as the tube diameter increase, the void fraction at the tube wall also increase (and thus the wall temperature) [63]. Consequently, smaller tube diameters are associated with higher CHF values. This expected behavior is correctly reflected in the PINN plot.
- **Pressure**: as pressure increases, also the CHF is expected to increase due to a higher boiling point. However, once the critical pressure is reached (around 22 MPa for water), the CHF decreases with further pressure increase. Since the pressure ranges considered in this study don't go near the critical pressure, the NN pressure plot is the accurate representation of the physical reality.
- **Outlet quality**: an increase in outlet quality translates in higher void fractions which leads to less water in direct contact with the heated wall and thereby

lower heat exchange. With increasing outlet quality the CHF is therefore expected to decrease to a lower value, a trend shown in the PINN result.

Overall, incorporating a physical correlation through a PINN framework results in a predictive behavior more consistent with the expected physical behavior, with the exception, for this specific application, of pressure. While PINN captures a correct qualitative behavior, it's quantitative accuracy, as highlighted by the results presented in Chapter 5.2, make it underperforming with respect to NN models.

Lastly, it's addressed the quantitative analysis performed with LIME by extracting the maximum values of the importance (both positive and negative contributions) for each addressed feature. The importance value assigned to a feature represents the degree and direction of that feature's impact on the model's final prediction for a specific instance. In LIME, it represents the relative contribution of a feature to the output prediction in a local context. A higher absolute value of importance indicates that the feature considered has a strong influence on the predicted output. Positive importance values mean that an increase in this feature also increases the output value (higher CHF), while negative importance values indicate that an increase in the feature decreases the output value (lower CHF). All results are collected in Table 6.1 and outlet quality and heated length result as the most influential parameters for the PINN, while heated length and mass flux do for the NN. This difference could be a contributing factor to the performance discrepancy between the two models.

It's important to clearly state the limitations of this approach: LIME only approximates the model locally and thus it may not generalize well to other data points, it's quite simple and as such it may not capture the full complexity of the NN's global behavior and data point are locally created without considering the eventual correlations between features which can lead to unrealistic data points

Table 6.1: Importance values for Individual NN and PINN models

Input Parameter	**Individual NN Model**		**PINN Model**	
	Max Positive Importance	*Min Negative Importance*	*Max Positive Importance*	*Min Negative Importance*
Tube Diameter	0.076	-0.070	0.015	-0.024
Heated Length	0.231	-0.165	0.070	-0.069
Pressure	0.042	-0.050	0.064	-0.058
Mass Flux	0.223	-0.178	0.042	-0.051
Outlet Quality	0.122	-0.101	0.108	-0.075

6.2. SHAP

SHAP (SHapley Additive exPlanations) is an approach that explains the output of any machine learning model using game theory, particularly the Shapley value. The Shapley values capture the average contribution of each feature by considering all possible combinations of features. In the context of SHAP, each feature in a complex model is considered a "contributor" to the prediction, with Shapley values quantifying each feature's impact on the final outcome [64]. SHAP calculates these values by assessing how the model's output changes when conditioned on informations from each feature, averaging this impact across all possible feature combinations.

LIME interprets individual model predictions based on local approximations of the model around a given prediction (instance) and thus it's better suited for localized insights and simple models. If the interest is to gain a broader understanding on the model behavior, involving both global and local perspectives, and concerns complex models, SHAP is more suitable. Also, LIME might show inconsistent results across runs due to random sampling while SHAP displays more consistent and stable results. For these reasons both approaches have been investigated. As for LIME, both PINN and NN are analyzed with SHAP and in Figure 6.2 and Figure 6.3 the overall importance of each feature is shown.

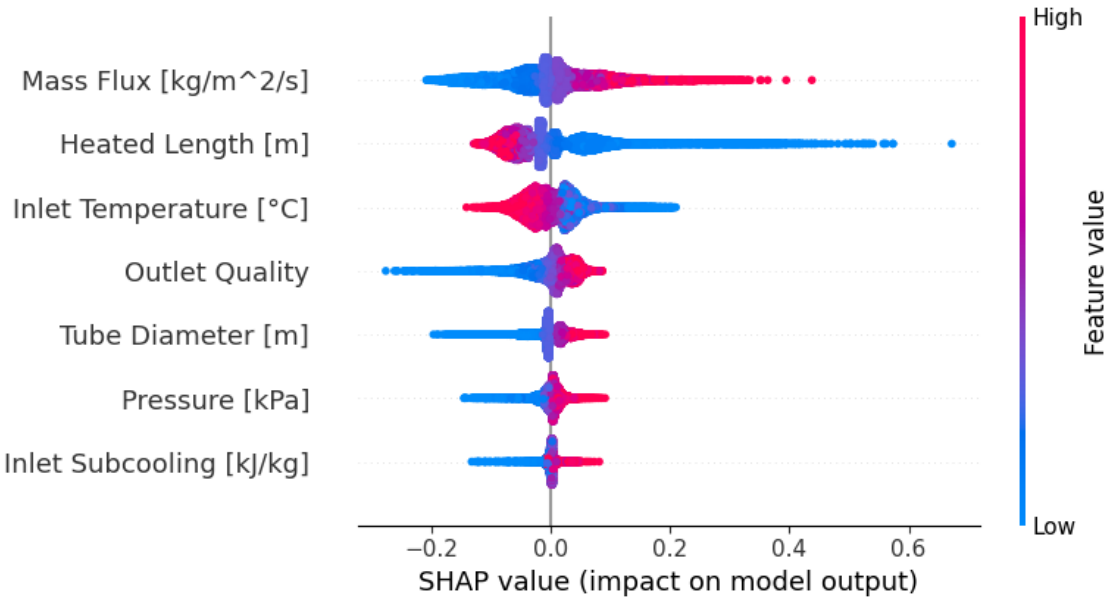


Figure 6.2: SHAP values per input parameter for the NN model

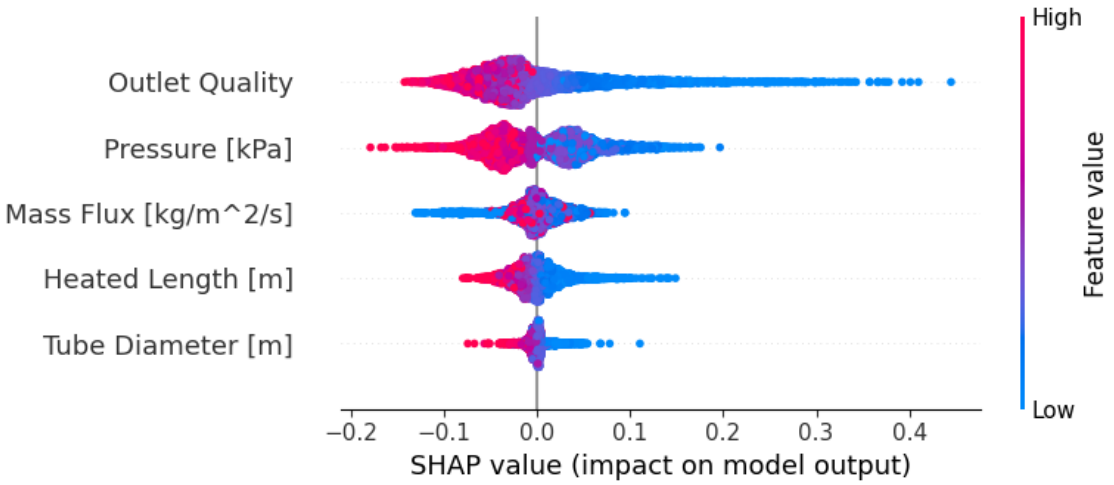


Figure 6.3: SHAP values per input parameter for the PINN model

The plots in Figure 6.2 and Figure 6.3 provide an overview of which features are most important for the considered model. Those plots sort features by the sum of SHAP value magnitudes over all samples (all rows of the 80% train-validation dataset), with the most significant one on top and the less important following. By looking at those results, for PINN the most important features are outlet quality and pressure, and for the NN mass flux and heated length resulted as the features influencing the output the most. Both results are coherent with the results from LIME. The color represents the feature value, for example for high values of outlet quality (red), negative SHAP values are obtained, meaning the CHF decreases for those values; while for low values (blue), positive SHAP values are registered and thus CHF increases.

To understand how each single feature affects the output of the model, dependence scatter plot are created considering the PINN model and are presented in Figure 6.4. In each plot, SHAP values considering one feature at a time are plotted vs. the feature's values. Since SHAP values represent a feature's responsibility for a change in the model output, those plots represent how CHF changes as each feature changes. Vertical dispersion represents interaction effects with other features, specifically SHAP picks the feature interacting the most with the considered one.

The partial dependence plots collected show the same behavior observed with LIME, both for the PINN model and NN model, here for simplicity and relevance, only the PINN results are reported. The same observations noted in Section 6.1 can be derived also now, with the added consideration on features interactions. Focusing again on outlet quality, mass flux is indicated as the parameter interacting with it most relevantly; it also shows that for low outlet quality values, high mass fluxes (red) lead to higher SHAP values and thus higher CHF.

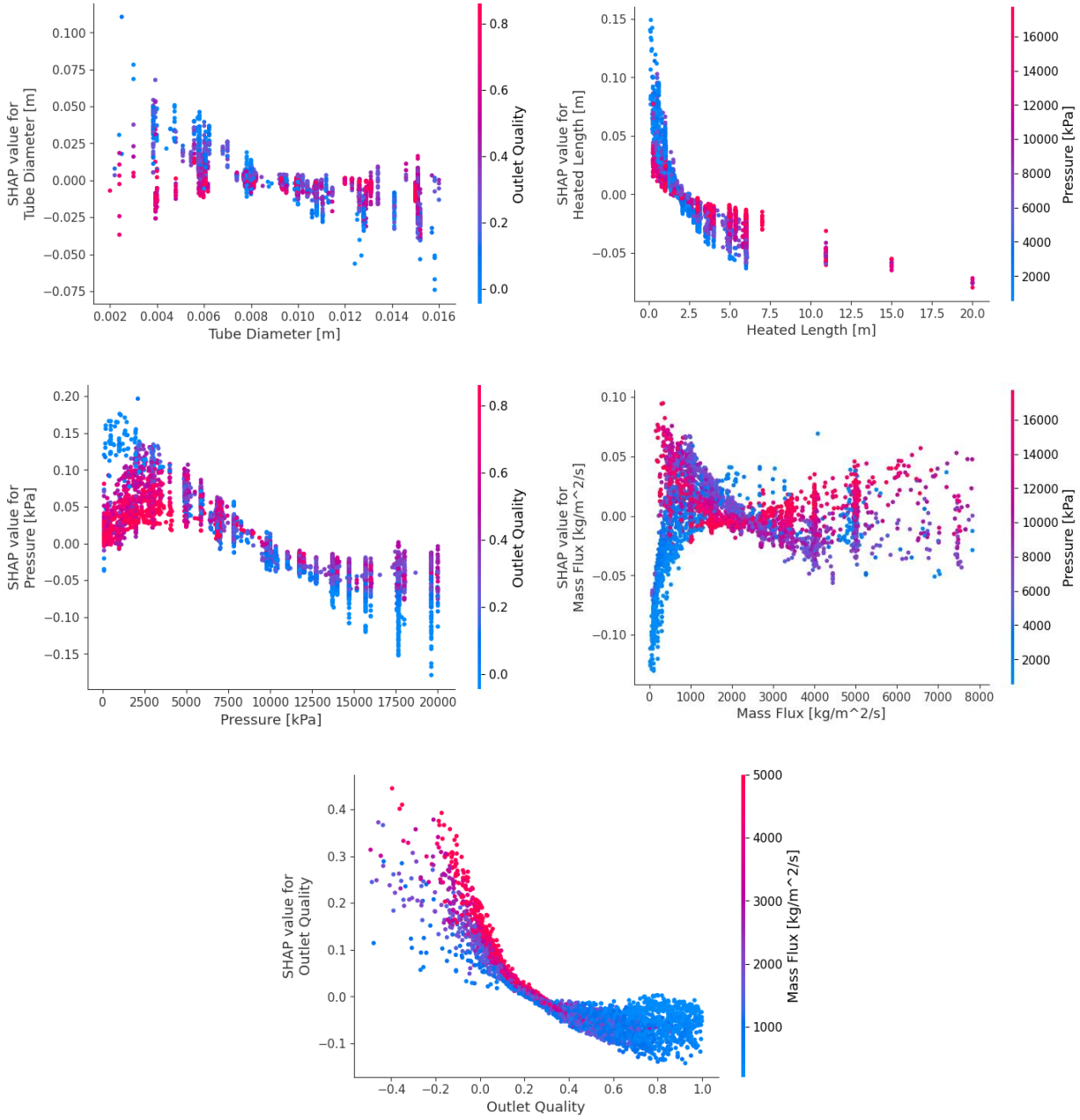


Figure 6.4: SHAP partial dependence plots per feature

7 Conclusion and Future Developments

This work presents new ML-based methods for CHF prediction in vertical tubes to enhance predictive accuracy: an ensemble of optimized neural networks and a physics-informed neural network. First, multiple individual NN models are developed, each optimally trained with distinct architectures and hyperparameters settings. These individual models are, then, combined to form an optimized ensemble model. Two systematic procedures are introduced: 1) a procedure for automatically identifying optimal NN models; and 2) a procedure for aggregating optimized individual NN models into an optimal ensemble model. The proposed procedures allow the identification of the optimal ensemble configuration without relying on computationally intensive optimization techniques. Second, a physics-informed neural network is developed starting from a 5 input NN model as baseline and integrating in its learning process the Westinghouse empirical correlation. The proposed methods are validated using the experimental CHF data from the WPRS-EGMUP task force on AI and ML for Scientific Computing in Nuclear Engineering projects, promoted by the OECD/NEA.

The obtained results for the ensemble of NNs model show that the proposed method offers superior accuracy in CHF prediction, achieving lower errors across all metrics considered (MAPE= 2.04, RMSPE= 5.94) when compared to conventional methods such as the LUT approach (MAPE= 22.31, RMSPE= 43.05) and other ML techniques like SVR (MAPE= 8.03, RMSPE= 15.53) and RF (MAPE= 4.64, RMSPE= 7.97). The optimized ensemble model achieves a coverage of 99.55% of predicted data points within a ±20% error margin, marking a significant improvement over conventional techniques (LUT achieves only 67.05%). Parametric and sensitivity analyses further confirm the strong generalization capability of the proposed method and its consistency with expected physical behaviors, thereby reinforcing the effectiveness of the proposed method for CHF prediction.

The second approach investigated is the PINN. Few limitations have been detected from the beginning, ranging from the unavailability of PDEs to the high uncertainties of the selected correlation. This leads to a worst performance of the PINN class of ML methods with respect to the simple NNs implementation. This result is probably due to the misleading informations provided by the W-3 physical correlation because of its high inaccuracy. Out of the two approaches investigated for the physics loss function

definition, the simple difference shows the best results, however not significantly better than NNs'. The results for an optimized lambda value of 0.04 indicates a RMSPE of 17.00 and only an 88.16% of datapoints within a ±20% error margin. Also, for PINN, 5-input models have been trained and tested to check for any redundancy in the informations provided to the NN which might lead to overfitting. As a result, no proof of worst generalization capabilities are detected and exploiting all available inputs from the NRC CHF dataset is strongly suggested for better performance.

Finally, understanding why a model makes a certain prediction can be as crucial as the prediction's accuracy and as such, two different interpretability methods, namely LIME and SHAP, have been used to handle this problem. The results show that the physical behavior has been correctly implemented in the NN through the introduction of the W-3 correlation in the physics loss for the total loss function. However, the high uncertainty in the correlation accuracy leads to worst quantitative results than those of the NNs trained only through supervised learning.

Starting from these considerations, different results are not expected by implementing similar empirical correlations in the ML framework through PINN. However, more updated correlations and methods for CHF prediction, which make use of empirical parameters based on up-to-date experimental datasets and latest phenomenological researches (lower expected uncertainty given the larger set of informations), could provide an interesting starting point for future PINN development. From the scatter plots of the input data distributions it's also evidenced which are the regions for each parameter with less experimental data available. From this considerations, future experimental work should fill those gaps in order to have more accurate insights into how each parameter influences the CHF. Lastly, ML models have all shown higher CHF prediction accuracy with respect to LUTs and CHF correlations, specifically NN have proven superior effectiveness and should be further researched for tackling the CHF prediction problem.

Bibliography

[1] L. S. Tong and Y. S. Tang, *Boiling Heat Transfer and Two-Phase Flow*, 2nd ed. Routledge, 2018. doi: 10.1201/9781315138510.

[2] S. Chang and W.-P. Baek, "Understanding, predicting, and enhancing critical heat flux", the 10th Int. Topical Meeting on Nuc. Reactor Thermal Hydraulics (NURETH-10), Seoul, Korea, October 5-9, 2003.

[3] N. E. Todreas and M. Kazimi, *Nuclear Systems Volume I*, 0 ed. CRC Press, 2011. doi: 10.1201/b14887.

[4] G. P. Celata, M. Cumo, A. Mariani, M. Simoncini, and G. Zummo, "Rationalization of existing mechanistic models for the prediction of water subcooled flow boiling critical heat flux," *International Journal of Heat and Mass Transfer*, vol. 37, pp. 347–360, Mar. 1994, doi: 10.1016/0017-9310(94)90035-3.

[5] M. Bruder, G. Bloch, and T. Sattelmayer, "Critical Heat Flux in Flow Boiling—Review of the Current Understanding and Experimental Approaches," *Heat Transfer Engineering*, vol. 38, no. 3, pp. 347–360, Feb. 2017, doi: 10.1080/01457632.2016.1189274.

[6] D. C. Groeneveld, "Critical heat flux data used to generate the 2006 Groeneveld critical heat flux lookup tables." Washington, DC: United States Nuclear Regulatory Commission, Office of Nuclear Regulatory Research, 2019.

[7] E. Grosfilley, G. Robertson, J. Soibam, and J.-M. Corre, "Investigation of Machine Learning Regression Techniques to Predict Critical Heat Flux over a Large Parameter Space," in *20th International Topical Meeting on Nuclear Reactor Thermal Hydraulics (NURETH-20)*, Washington, D.C.: American Nuclear Society, 2023, pp. 4516–4529. doi: 10.13182/NURETH20-39985.

[8] X. Zhao, K. Shirvan, R. K. Salko, and F. Guo, "On the prediction of critical heat flux using a physics-informed machine learning-aided framework," *Applied Thermal Engineering*, vol. 164, p. 114540, Jan. 2020, doi: 10.1016/j.applthermaleng.2019.114540.

[9] E. Schiassi, M. De Florio, B. D. Ganapol, P. Picca, and R. Furfaro, "Physics-informed neural networks for the point kinetics equations for nuclear reactor dynamics," *Annals of Nuclear Energy*, vol. 167, p. 108833, Mar. 2022, doi: 10.1016/j.anucene.2021.108833.

[10] J.-L. Wu, H. Xiao, and E. Paterson, "Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework," *Phys. Rev. Fluids*, vol. 3, no. 7, p. 074602, Jul. 2018, doi: 10.1103/PhysRevFluids.3.074602.

[11] U. Forssell and P. Lindskog, "Combining Semi-Physical and Neural Network Modeling: An Example ofIts Usefulness," *IFAC Proceedings Volumes*, vol. 30, no. 11, pp. 767–770, Jul. 1997, doi: 10.1016/S1474-6670(17)42938-7.

[12] E. Zhang, M. Dao, G. E. Karniadakis, and S. Suresh, "Analyses of internal structures and defects in materials using physics-informed neural networks," *Sci. Adv.*, vol. 8, no. 7, p. eabk0644, Feb. 2022, doi: 10.1126/sciadv.abk0644.

[13] D. Jalili, S. Jang, M. Jadidi, G. Giustini, A. Keshmiri, and Y. Mahmoudi, "Physics-informed neural networks for heat transfer prediction in two-phase flows," *International Journal of Heat and Mass Transfer*, vol. 221, p. 125089, Apr. 2024, doi: 10.1016/j.ijheatmasstransfer.2023.125089.

[14] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis, "Physics-Informed Neural Networks for Heat Transfer Problems," *Journal of Heat Transfer*, vol. 143, no. 6, p. 060801, Jun. 2021, doi: 10.1115/1.4050542.

[15] Q. Huang *et al.*, "A review of the application of artificial intelligence to nuclear reactors: Where we are and what's next," *Heliyon*, vol. 9, no. 3, p. e13883, Mar. 2023, doi: 10.1016/j.heliyon.2023.e13883.

[16] LE CORRE, Jean-Marie, et al., "Benchmark on Artificial Intelligence and Machine Learning for Scientific Computing in Nuclear Engineering. Phase 1: Critical Heat Flux Exercise Specifications," vol. OECD Publishing, Paris, no. NEA Working Papers, 2024.

[17] M. C. Vlachou, J. S. Lioumbas, and T. D. Karapantsios, "HEAT TRANSFER ENHANCEMENT IN BOILING OVER MODIFIED SURFACES: A CRITICAL REVIEW," *Interfac Phenom Heat Transfer*, vol. 3, no. 4, pp. 341–367, 2015, doi: 10.1615/InterfacPhenomHeatTransfer.2016014133.

[18] S. Dong, S. Gong, B. Zhang, Y. Yuan, and W. Ma, "Mechanistic critical heat flux model development for subcooled flow boiling based on superheated liquid sublayer depletion," *Progress in Nuclear Energy*, vol. 153, p. 104445, Nov. 2022, doi: 10.1016/j.pnucene.2022.104445.

[19] C. H. Lee and I. Mudawar, "A mechanistic critical heat flux model for subcooled flow boiling based on local bulk flow conditions," *International Journal of Multiphase Flow*, vol. 14, no. 6, pp. 711–728, Nov. 1988, doi: 10.1016/0301-9322(88)90070-5.

[20] F. T. Kanizawa and G. Ribatski, "Critical Heat Flux and Dryout," in *Flow boiling and condensation in microscale channels*, in Mechanical Engineering Series. , Cham: Springer International Publishing, 2021, pp. 217–240. doi: 10.1007/978-3-030-68704-5_6.

[21] L. S. Tong, "Prediction of departure from nucleate boiling for an axially non-uniform heat flux distribution," *Journal of Nuclear Energy*, vol. 21, no. 3, pp. 241–248, Jan. 1967, doi: 10.1016/S0022-3107(67)90054-8.

[22] L. Biasi, G. C. Clerici, A. Tozzi, and R. Sala, "Extension of A.R.S. correlation to burnout prediction with non-uniform heating," *Journal of Nuclear Energy*, vol. 22, no. 12, pp. 705–716, Dec. 1968, doi: 10.1016/0022-3107(68)90044-0.

[23] E. D. Hughes, "A CORRELATION OF ROD BUNDLE CRITICAL HEAT FLUX FOR WATER IN THE PRESSURE RANGE 150 TO 725 PSIA.," IN--1412, 4093330, Jan. 1970. doi: 10.2172/4093330.

[24] R. W. Bowring, "WSC-2: a subchannel dryout correlation for water-cooled clusters over the pressure range 3.4-15.9 MPA (500-2300 PSIA)." Atomic Energy Establishment Winfrith, 1973.

[25] L. Levitan, L.L. F. P., "Investigating burnout with flow of a steam-water mixture in a round tube," *Teploenergetika*, vol. 22 (1), pp. 80–83.

[26] P. B. Whalley, "The calculation of dryout in a rod bundle," *International Journal of Multiphase Flow*, vol. 3, no. 6, pp. 501–515, Dec. 1977, doi: 10.1016/0301-9322(77)90026-X.

[27] J. Weisman and B. S. Pei, "Prediction of critical heat flux in flow boiling at low qualities," *International Journal of Heat and Mass Transfer*, vol. 26, no. 10, pp. 1463–1477, Oct. 1983, doi: 10.1016/S0017-9310(83)80047-7.

[28] Soon Heung Chang and Kwang Won Lee, "A critical heat flux model based on mass, energy, and momentum balance for upflow boiling at low qualities," *Nuclear Engineering and Design*, vol. 113, no. 1, pp. 35–50, Apr. 1989, doi: 10.1016/0029-5493(89)90294-X.

[29] W.-S. Lin, C.-H. Lee, and B.-S. Pei, "An Improved Theoretical Critical Heat Flux Model for Low-Quality Flow," *Nuclear Technology*, vol. 88, no. 3, pp. 294–306, Dec. 1989, doi: 10.13182/NT89-A34312.

[30] Y. Katto, "A prediction model of subcooled water flow boiling CHF for pressure in the range 0.1–20 MPa," *International Journal of Heat and Mass Transfer*, vol. 35, no. 5, pp. 1115–1123, May 1992, doi: 10.1016/0017-9310(92)90172-O.

[31] G. P. Celata, M. Cumo, Y. Katto, and A. Mariani, "Prediction of the critical heat flux in water subcooled flow boiling using a new mechanistic approach," *International Journal of Heat and Mass Transfer*, vol. 42, no. 8, pp. 1457–1466, Apr. 1999, doi: 10.1016/S0017-9310(98)00286-5.

[32] Y. M. Kwon and S. H. Chang, "A mechanistic critical heat flux model for wide range of subcooled and low quality flow boiling," *Nuclear Engineering and Design*, vol. 188, no. 1, pp. 27–47, Apr. 1999, doi: 10.1016/S0029-5493(99)00025-4.

[33] I. Kataoka, M. Ishii, and A. Nakayama, "Entrainment and desposition rates of droplets in annular two-phase flow," *International Journal of Heat and Mass Transfer*, vol. 43, no. 9, pp. 1573–1589, May 2000, doi: 10.1016/S0017-9310(99)00236-7.

[34] W. Liu, "Prediction of Critical Heat Flux for Subcooled Flow Boiling in Annulus and Transient Surface Temperature Change at CHF," *Fluids*, vol. 7, no. 7, p. 230, Jul. 2022, doi: 10.3390/fluids7070230.

[35] X. Cheng and U. Muller, "Review on critical heat flux in water cooled reactors," *FZKA*, 2003.

[36] J. E. Dahlquist, F. S. Gl, and R. A. Nelson, "Considerations for Modeling Critical Heat Flux Behavior," *Nuclear Technology*, vol. 68, no. 2, pp. 252–262, Feb. 1985, doi: 10.13182/NT85-A33558.

[37] V. E. Doroshchuk, I. L. Levitan, and F. P. Lantzman, "Investigation into Burnout in Uniformly Heated Tubes," *ASME Publication*, vol. 75-WA/HT-22, 1975.

[38] D. C. Groeneveld *et al.*, "The 1995 look-up table for critical heat flux in tubes," *Nuclear Engineering and Design*, vol. 163, no. 1–2, pp. 1–23, Jun. 1996, doi: 10.1016/0029-5493(95)01154-4.

[39] D. C. Groeneveld *et al.*, "The 2006 CHF look-up table," *Nuclear Engineering and Design*, vol. 237, no. 15–17, pp. 1909–1922, Sep. 2007, doi: 10.1016/j.nucengdes.2007.02.014.

[40] A. Mazzola, "Integrating artificial neural networks and empirical correlations for the prediction of water-subcooled critical heat flux," *Revue Générale de Thermique*, vol. 36, no. 11, pp. 799–806, Dec. 1997, doi: 10.1016/S0035-3159(97)87750-1.

[41] M. R. Lee, H.-J. Jeong, Y. J. Choi, and T. M. Gatton, "A Nuclear Power Plant Expert System Using Artificial Neural Networks," in *Advances in Neural Networks - ISNN 2006*, vol. 3972, J. Wang, Z. Yi, J. M. Zurada, B.-L. Lu, and H. Yin, Eds., in Lecture Notes in Computer Science, vol. 3972. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1239–1245. doi: 10.1007/11760023_180.

[42] B. T. Jiang and F. Y. Zhao, "Combination of support vector regression and artificial neural networks for prediction of critical heat flux," *International Journal of Heat and Mass Transfer*, vol. 62, pp. 481–494, Jul. 2013, doi: 10.1016/j.ijheatmasstransfer.2013.03.025.

[43] M. He and Y. Lee, "Application of machine learning for prediction of critical heat flux: Support vector machine for data-driven CHF look-up table construction based on sparingly distributed training data points," *Nuclear Engineering and Design*, vol. 338, pp. 189–198, Nov. 2018, doi: 10.1016/j.nucengdes.2018.08.005.

[44] R. Z. Khalid, A. Ullah, A. Khan, A. Khan, and M. H. Inayat, "Comparison of Standalone and Hybrid Machine Learning Models for Prediction of Critical Heat

Flux in Vertical Tubes," *Energies*, vol. 16, no. 7, p. 3182, Mar. 2023, doi: 10.3390/en16073182.

[45] C. Mao and Y. Jin, "Uncertainty quantification study of the physics-informed machine learning models for critical heat flux prediction," *Progress in Nuclear Energy*, vol. 170, p. 105097, May 2024, doi: 10.1016/j.pnucene.2024.105097.

[46] R. Zubair Khalid, I. Ahmed, A. Ullah, E. Zio, and A. Khan, "Enhancing accuracy of prediction of critical heat flux in Circular channels by ensemble of deep sparse autoencoders and deep neural Networks," *Nuclear Engineering and Design*, vol. 429, p. 113587, Dec. 2024, doi: 10.1016/j.nucengdes.2024.113587.

[47] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943, doi: 10.1007/BF02478259.

[48] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech Recognition Using Deep Neural Networks: A Systematic Review," *IEEE Access*, vol. 7, pp. 19143–19165, 2019, doi: 10.1109/ACCESS.2019.2896880.

[49] K.-L. Du, "Clustering: A neural network approach," *Neural Networks*, vol. 23, no. 1, pp. 89–107, Jan. 2010, doi: 10.1016/j.neunet.2009.08.007.

[50] "What is a Neural Network? | IBM." Accessed: Jun. 17, 2024. [Online]. Available: https://www.ibm.com/topics/neural-networks

[51] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986, doi: 10.1038/323533a0.

[52] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, no. 10, pp. 993–1001, Oct. 1990, doi: 10.1109/34.58871.

[53] R. Wilson, *Combinatorics: A Very Short Introduction*, 1st ed. Oxford University Press, 2016. doi: 10.1093/actrade/9780198723493.001.0001.

[54] M. Raissi, P. Perdikaris, and G. E. 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, pp. 686–707, Feb. 2019, doi: 10.1016/j.jcp.2018.10.045.

[55] 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, Feb. 2021, doi: 10.1016/j.jcp.2020.109951.

[56] Y. Diao, J. Yang, Y. Zhang, D. Zhang, and Y. Du, "Solving multi-material problems in solid mechanics using physics-informed neural networks based on

domain decomposition technology," *Computer Methods in Applied Mechanics and Engineering*, vol. 413, p. 116120, Aug. 2023, doi: 10.1016/j.cma.2023.116120.

[57] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," 2015, doi: 10.48550/ARXIV.1502.05767.

[58] A. Bhandari, "Feature Scaling: Engineering, Normalization, and Standardization (Updated 2024)," Analytics Vidhya. Accessed: Jun. 20, 2024. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/

[59] S. Sharma, S. Sharma, and A. Athaiya, "ACTIVATION FUNCTIONS IN NEURAL NETWORKS," *IJEAST*, vol. 04, no. 12, pp. 310–316, May 2020, doi: 10.33564/IJEAST.2020.v04i12.054.

[60] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," 2015, *arXiv*. doi: 10.48550/ARXIV.1502.01852.

[61] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2014, *arXiv*. doi: 10.48550/ARXIV.1412.6980.

[62] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why Should I Trust You?': Explaining the Predictions of Any Classifier," 2016, *arXiv*. doi: 10.48550/ARXIV.1602.04938.

[63] "The Effect of Linear Change of Tube Diameter on Subcooled Flow Boiling and Critical Heat Flux," *IJE*, vol. 33, no. 8, Aug. 2020, doi: 10.5829/ije.2020.33.08b.17.

[64] S. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," 2017, *arXiv*. doi: 10.48550/ARXIV.1705.07874.

A Appendix A

In this Appendix section, the NN architectures for Model A and for the Ensemble are presented in section A.1 and the general set ups for metrics evaluation and parity plot construction are shown in section A.2. Model A has shown the best statistical metrics results throughout the individual models with seven input parameters analyzed. All other presented models follow the same structure as the one described below. The only adjustments to make pertain to the number of nodes and hidden layers by adding or removing self. rows in the code and those of the corresponding weights and bias initialization, as well as in the forward definition step.

Section A.2 reports the metrics calculations and plots generation. The reported code section is taken from the testing of the performance of the various individual neural network models. This framework has then been reused, with the necessary adjustments, in other contexts where similar graphs and results are produced and have been presented in this work.

A.1. Neural Network Architecture

Algorithm 1 Architecture_Model_A

```
# %% MODEL A

class CHF_nnModel_A (nn.Module) :
    def __init__(self) :
        super(CHF_nnModel_A, self).__init__()
        self.input        = nn.Linear(7, 70)
        self.act1         = nn.ReLU()
        self.dropout1     = nn.Dropout(0)

        self.hidden1      = nn.Linear(70, 60)
        self.act2         = nn.ReLU()
        self.dropout2     = nn.Dropout(0)

        self.output       = nn.Linear(60, 1)
        self.act_output = nn.Softplus(beta=4)

        # He weights initialization
```

```
18:         torch.manual_seed(42)
19:         nn.init.kaiming_normal_(self.input.weight, mode='fan_in',
                                    nonlinearity='relu')
20:         nn.init.kaiming_normal_(self.hidden1.weight, mode='fan_in',
                                     nonlinearity='relu')
21:         nn.init.kaiming_normal_(self.output.weight, mode='fan_in',
                                     nonlinearity='relu')
22:
23:         # bias initialization
24:         nn.init.zeros_(self.input.bias)
25:         nn.init.zeros_(self.hidden1.bias)
26:         nn.init.zeros_(self.output.bias)
27:
28:       def forward(self, x_train) :
29:         out = self.act1(self.input(x_train))
30:         out = self.dropout1(out)
31:         out = self.act2(self.hidden1(out))
32:         out = self.dropout2(out)
33:         out = self.act_output(self.output(out))
34:         return out
35:
36:   model = CHF_nnModel_A().to(torch.float64)    # write this when calling
                                                     the above nn architecture
```

Algorithm 2 Architecture_Ensemble_Model

```
1:    # %% ENSEMBLE MODEL – A,B,D,E
2:
3:    def ensemble_model(input_data):
4:       # Load models
5:       model_A = CHF_nnModel_A().to(torch.float64)
6:       model_A.load_state_dict(torch.load('model_CHF_A.pt'))
7:       model_A.eval()
8:
9:       model_B = CHF_nnModel_B().to(torch.float64)
10:      model_B.load_state_dict(torch.load('model_CHF_B.pt'))
11:      model_B.eval()
12:
13:      model_D = CHF_nnModel_D().to(torch.float64)
14:      model_D.load_state_dict(torch.load('model_CHF_D.pt'))
15:      model_D.eval()
16:
```

```
17:     model_E = CHF_nnModel_E().to(torch.float64)
18:     model_E.load_state_dict(torch.load('model_CHF_E.pt'))
19:     model_E.eval()
20:
21:     # Output definition
22:     y_test_A  = model_A(input_data).detach().numpy()
23:     y_test_B  = model_B(input_data).detach().numpy()
24:     y_test_D  = model_D(input_data).detach().numpy()
25:     y_test_E  = model_E(input_data).detach().numpy()
26:
27:     combined_output = np.mean([y_test_A, y_test_B, y_test_D, y_test_E],
                                  axis=0)
28:
29:     return combined_output
30:
31: # to use this :
32: # CHF_ens = ensemble_model(put here the input matrix)
```

A.2. Testing Metrics and Plots

Algorithm 3 Test_Metrics_and_Plots

```
1:  # import libraries
2:
3:  import torch
4:  from torch import nn
5:  import torch.optim as optim
6:  import pandas as pd
7:  import numpy as np
8:  from sklearn.model_selection import train_test_split
9:  from sklearn.preprocessing import MinMaxScaler
10: from sklearn.metrics import mean_absolute_error
11: from sklearn.metrics import root_mean_squared_error
12: from sklearn.model_selection import KFold
13: import matplotlib.pyplot as plt
14: import CHF_nnArchitectures
15:
16: # DATA MANAGEMENT
17:
18: db                                                              =
    pd.read_csv('C:\\Users\\irene\Desktop\\THESIS\\CHF_Dataset_Excel
                \\chf_public_LUT.csv', delimiter = ',')
19:
```

```
20:    drop = db.drop_duplicates()
21:    data = drop.iloc[1: , 2:]
22:
23:    # Dataset split
24:    x = data.iloc[0: , 0:7].values
25:    y = data.iloc[0: , 7:].values
26:
27:    x_new , x_test , y_NEW , y_TEST = train_test_split(x,y, test_size=0.2,
                                                          random_state=1)
28:
29:    y_new  = y_NEW[0: , -2]
30:    y_new = y_new.reshape(len(y_new),1)
31:    y_test = y_TEST[0: , -2]
32:    y_test = y_test.reshape(len(y_test),1)
33:
34:    LUT_test = y_TEST[0: , -1]
35:    LUT_test = LUT_test.reshape(len(LUT_test),1)
36:
37:    # EVALUATION ON TEST SET AND LUT
38:
39:    scaler_in = MinMaxScaler()
40:    scaler_in.fit(x_new)
41:    scaler_out = MinMaxScaler()
42:    scaler_out.fit(y_new)
43:
44:    # Scaling
45:    x_test = scaler_in.transform(x_test)
46:    y_test = scaler_out.transform(y_test)
47:    LUT_scaled = scaler_out.transform(LUT_test)
48:
49:    # Tensor
50:    x_test = torch.tensor(x_test.astype(np.float64))
51:    y_test = torch.tensor(y_test.astype(np.float64))
52:
53:    # Delete zeros
54:    # x_new, x_test, y_new, y_test, LUT_scaled have all been checked
55:    # only y_new showed one row with 0 value to be eliminated
56:
57:    # Testing
58:    model = CHF_nnArchitectures.CHF_nnModel_A().to(torch.float64)
59:    model.load_state_dict(torch.load('model_CHF_A.pt'))
60:    model.eval()
```

```
61:
62:     y_test_pred = model(x_test)
63:     y_test_pred_np = y_test_pred.detach().numpy()
64:
65:     y_test_np = y_test.detach().numpy()
66:
67:     RMSPE = 100*np.sqrt(np.sum(np.square((y_test_pred_np –
                                  y_test_np)/y_test_np))/len(y_test_np))
68:     MAPE   = 100*np.sum(np.abs((y_test_pred_np –
                                  y_test_np)/y_test_np))/len(y_test_np)
69:     Q2         = np.sum(np.square(y_test_pred_np –
                              y_test_np))/np.sum(np.square(y_test_pred_np –
                              np.mean(y_test_pred_np)))
70:
71:     print(f'RMSPE - Model : {RMSPE}\nMAPE - Model : {MAPE}\nQ2 - Model :
            {Q2}\n----------')
72:
73:     RMSPE_LUT = 100*np.sqrt(np.sum(np.square((LUT_scaled –
                                        y_test_np)/y_test_np))/len(y_test_np))
74:     MAPE_LUT  = 100*np.sum(np.abs((LUT_scaled –
                                        y_test_np)/y_test_np))/len(y_test_np)
75:     Q2_LUT        = np.sum(np.square(LUT_scaled–y_test_np))/np.sum(
                                  np.square(LUT_scaled-np.mean(LUT_scaled)))
76:
77:     print(f'RMSPE - LUT : {RMSPE_LUT}\nMAPE - LUT : {MAPE_LUT}\nQ2 –
            LUT : {Q2_LUT}\n----------')
78:
79:     # PLOTS
80:
81:     # Predicted vs Measured
82:
83:     residuals = y_test_np - y_test_pred_np
84:     errors_percentage = np.abs(residuals/y_test_np * 100)
85:     error_a = 10
86:     error_b = 20
87:
88:     points_in_a = np.sum(errors_percentage <= error_a)
89:     percentage_points_a = (points_in_a/len(errors_percentage))*100
90:     print(f'The {percentage_points_a:.2f}% of predicted CHF values falls in
              ±{error_a}% error limit compared to measured values')
91:
92:     points_in_b = np.sum(errors_percentage <= error_b)
```

```
93:   percentage_points_b = (points_in_b/len(errors_percentage))*100
94:   print(f'The {percentage_points_b:.2f}% of predicted CHF values falls in
          ±{error_b}% error limit compared to measured values')
95:
96:   y_test_original = scaler_out.inverse_transform(y_test_np)
97:   y_test_pred_original = scaler_out.inverse_transform(y_test_pred_np)
98:
99:
100:  plt.scatter(y_test_pred_original, y_test_original, color ='tab:blue')
101:  plt.plot([min(y_test_pred_original),max(y_test_pred_original)],
              [min(y_test_pred_original),max(y_test_pred_original)],
102:          linestyle ='--', color ='k', label ='perfect match')
103:
104:  x_values = np.array([min(y_test_pred_original),max(y_test_pred_original)])
105:  plt.plot(x_values, x_values * 1.10, color ='gray', linestyle ='--', label ='Upper
          error limit 10%')
106:  plt.plot(x_values, x_values * 0.90, color ='gray', linestyle ='--', label ='Lower
          error limit 10%')
107:
108:  plt.xlabel('Predicted CHF values [kW/m^2]')
109:  plt.ylabel('Measured CHF values [kW/m^2]')
110:  plt.title('Predicted vs Measured - Model A')
111:  plt.legend()
112:  plt.show()
113:
114:  # LUT vs Measured
115:
116:  error_LUT = np.abs((y_test_np-LUT_scaled)/y_test_np * 100)
117:
118:  limit_a = 10
119:  inside_a = np.sum(error_LUT <= limit_a)
120:  percentage_a = (inside_a/len(error_LUT))*100
121:  print(f'The {percentage_a:.2f}% of predicted CHF values falls in ±{limit_a}%
          error limit compared to measured values')
122:
123:  limit_b = 20
124:  inside_b = np.sum(error_LUT <= limit_b)
125:  percentage_b = (inside_b/len(error_LUT))*100
126:  print(f'The {percentage_b:.2f}% of predicted CHF values falls in ±{limit_b}%
          error limit compared to measured values')
127:
128:  LUT_original = scaler_out.inverse_transform(LUT_scaled)
```

```
129:
130:    plt.scatter(LUT_original, y_test_original, color ='tab:blue')
131:    plt.plot([min(LUT_original),max(LUT_original)],
                [min(LUT_original),max(LUT_original)],
132:            linestyle ='--', color ='k', label ='perfect match')
133:
134:    x_values = np.array([min(LUT_original),max(LUT_original)])
135:
136:    plt.plot(x_values, x_values * 1.10, color ='gray', linestyle ='--', label ='Upper
            error limit 10%')
137:    plt.plot(x_values, x_values * 0.90, color ='gray', linestyle ='--', label ='Lower
            error limit 10%')
138:    plt.plot(x_values, x_values * 1.30, color ='gray', linestyle ='-.', label ='Upper
            error limit 30%')
139:    plt.plot(x_values, x_values * 0.70, color ='gray', linestyle ='-.', label ='Lower
            error limit 30%')
140:
141:    plt.xlabel('LUT CHF values [kW/m^2]')
142:    plt.ylabel('Measured CHF values[kW/m^2]')
143:    plt.title('LUT vs Measured')
144:    plt.legend()
145:    plt.show()
```

B Appendix B

In the following sections are presented two of the most important algorithms defined in the process of evaluating PINN. First, the custom scaling is shown to display the MinMaxScaler-based definition. Second, the entire loss function definition is put in evidence for both approaches. They both start with the Westinghouse-3 definition with the coefficients modified to account for the SI unit of measures for simplicity and coherence with the results from the dataset; it follows the definition of the physics losses, as explained in Chapter 3.3, and lastly the total loss function is composed along with the ranges of validity of the physical correlation.

B.1. Custom Scaling – MinMaxTensor

Algorithm 4 MinMaxTensor

```
# MinMaxTensor
#
# INPUT: tensor
# OUTPUT: scaled tensor

# %%
import torch

# %%
class MinMaxTensor :
    def __init__(self, min_value = 0.0, max_value = 1.0) :

        # This part keeps the max and min calculated during fit
        # and uses them to scale coherently new tensors
        self.min_value = min_value
        self.max_value = max_value
        self.data_min  = None
        self.data_max  = None

    def fit(self, tensor) :
        # Max and Min values of the tensor are calculated here
        #
```

```
23:         # dim=0 : calculates the max and min of each column moving along the
                      first dimension (rows)
24:         # keepdim=True : if the input is a matrix 2D m x n, the result will also be
                             a 2D vector 1 x n
25:         # .values : since the results are values and indices and we need just values
26:
27:         self.data_min = tensor.min(dim=0, keepdim=True).values
28:         self.data_max = tensor.max(dim=0, keepdim=True).values
29:
30:     def transform(self, tensor) :
31:         if self.data_min is None or self.data_max is None :
32:             raise ValueError('The scaler has not been fitted')
33:
34:         # Actual min-max scaling of the tensor
35:         scaled_tensor = (tensor - self.data_min) / (self.data_max - self.data_min)
36:         return scaled_tensor
37:
38:     def fit_transform(self, tensor) :
39:         # Combines fit and transform in a single passage
40:
41:         self.fit(tensor)
42:         return self.transform(tensor)
43:
44:     def inverse_transform(self, scaled_tensor):
45:         if self.data_min is None or self.data_max is None :
46:             raise ValueError('The scaler has not been fitted')
47:
48:         # Inverse scaling to get original values
49:         unscaled_tensor = scaled_tensor * (self.data_max - self.data_min) +
                                                 self.data_min
50:         return unscaled_tensor
```

B.2. PINN – Loss Function

Algorithm 5 Loss_Function_PINN_Simple_Difference_Method

```
1:  # LOSS FUNCTION
2:
3:  def westinghouse3_chf (D, P, G, X, H, scaler) :
4:
5:      # D : scaled column (tensor) - Diameter
6:      # P : scaled column (tensor) - Pressure
7:      # G : scaled column (tensor) - Mass Flux
```

```
8:          # X : scaled column (tensor) - Outlet Quality
9:          # H : scaled column (tensor) - Inlet Subcooling
10:         #
11:         # P is in [MPa] for Westinghouse, but in CHF dataset P is in [kPa]
            # thus *(10**-3)
12:         #
13:         # The model was trained with a matrix of scaled values
14:         # The loss cannot be computed on unscaled values and given as input
            # in loss.backward()
15:         # The would be computed in the wrong way, mismatch between scales
16:         # The result from the correlation will thus be scaled accordingly to
            # the output scaler of the problem
17:
18:         P_new = P * 1e-3
19:
20:         factor_1 = ((2.022 - 0.0624 * P_new) + (0.1722 - 0.0143 * P_new) *
                        torch.exp((18.177 - 0.599 * P_new) * X)) * (1.157 - 0.869 * X)
21:         factor_2 = (0.1484 - 1.596 * X + 0.1729 * X * torch.abs(X)) * 2.33 * G + 3271
22:         factor_3 = 0.2664 + 0.8357 * torch.exp(-124.01 * D)
23:         factor_4 = 0.8258 + 0.00034 * (H)
24:
25:         q_westing = (factor_1 * factor_2 * factor_3 * factor_4)
26:
27:         q_westing = q_westing.requires_grad_(True)
28:
29:         q_westing_sc = scaler.transform(q_westing)
30:
31:         return q_westing_sc
32:
33:
34:     def loss_total (model_CHF, true_CHF, correlation, D, P, G, X, lamb) :
35:
36:         # Parameters validity range for Westinghouse-3 correlation
37:         # Physics loss to be used only here
38:         D_top = 0.018
39:         D_bot = 0.005
40:
41:         P_top = 15860 #kPa
42:         P_bot = 6890  #kPa
43:
44:         G_top = 6781
45:         G_bot = 1356
```

```
46:
47:         X_top = 0.15
48:         X_bot = -0.15
49:
50:         validity_range = (
51:             (D_bot <= D) & (D <= D_top) &
52:             (P_bot <= P) & (P <= P_top) &
53:             (G_bot <= G) & (G <= G_top) &
54:             (X_bot <= X) & (X <= X_top)
55:         )
56:
57:         # Computation of the total_loss
58:         loss_data = (true_CHF - model_CHF) ** 2
59:         correlation = correlation.t() #transported vector ([1, 24] -> [24, 1]) to
                                          match model_CHF
60:         loss_phys = (model_CHF - correlation) ** 2
61:
62:         # print(f'loss data: {loss_data}')
63:         # print(f'loss_phys: {loss_phys}')
64:
65:         # Initialization loss_total vector for cycle
66:         loss_total = torch.zeros_like(loss_data)
67:
68:         for i in range(loss_data.size(0)) :
69:
70:             if validity_range[i].item() :
71:                 loss_total[i] = (1 - lamb) * loss_data[i] + lamb * loss_phys[i]
72:
73:             else :
74:                 loss_total[i] = loss_data[i]
75:
76:         loss = torch.mean(loss_total)
77:         # print(f'loss_mean: {loss}')
78:
79:         return loss, torch.mean(loss_data)
```

Algorithm 6 Loss_Function_PINN_Partial_Derivatives_Method

```
1:      # LOSS FUNCTION
2:
3:      def westinghouse3_chf (matrix_scaled, H, scaler_x, scaler_y) :
4:
```

```
5:      # DEFINITION INPUTS
6:      # 1.   matrix_scaled = [D, L, P, G, X]
7:      #                D : scaled column (tensor) - Diameter [m]
8:      #                L : scaled column (tensor) - Heated Length [m]
9:      #                P : scaled column (tensor) - Pressure [kPa]
10:     #                G : scaled column (tensor) - Mass Flux [kg/s/m^2]
11:     #                X : scaled column (tensor) - Outlet Quality
12:     #
13:     # 2.   H : Inlet Subcooling (unscaled tensor column) [kJ/kg]
14:     #
15:     # 3.   scaler_x and scaler_y previously fitted scalers on matrix_unscaled and
        #      original output
16:     #
17:     # NOTES FOR USE
18:     # - P is needed in [MPa] for Westinghouse, thus *(10**-3)
19:     # - Cannot directly use unscaled inputs because the gradients are calculated
        #   on scaled values,
20:     #   hence the code need to see a direct connection to those
21:     # - The model is trained on scaled values and so the loss cannot be computed
        #   on unscaled values and given as input in loss.backward() -> scales
        #   unmatched -> need to scale correlation result (which is computed with
        #   unscaled, original values)
22:
23:     Ds = matrix_scaled[0: , 0]
24:     Ls = matrix_scaled[0: , 1]
25:     Ps = matrix_scaled[0: , 2]
26:     Gs = matrix_scaled[0: , 3]
27:     Xs = matrix_scaled[0: , 4]
28:
29:     scaled = torch.stack([Ds, Ls, Ps, Gs, Xs], dim=1)
30:     matrix_unscaled = scaler_x.inverse_transform(scaled)
31:
32:     D = matrix_unscaled[0: , 0]
33:     P = matrix_unscaled[0: , 2]
34:     G = matrix_unscaled[0: , 3]
35:     X = matrix_unscaled[0: , 4]
36:
37:     P_new = P * 1e-3
38:
39:     factor_1 = ((2.022 - 0.0624 * P_new) + (0.1722 - 0.0143 * P_new) *
                        torch.exp((18.177 - 0.599 * P_new) * X)) * (1.157 - 0.869 * X)
40:     factor_2 = (0.1484 - 1.596 * X + 0.1729 * X * torch.abs(X)) * 2.33 * G + 3271
```

```
41:       factor_3 = 0.2664 + 0.8357 * torch.exp(-124.01 * D)
42:       factor_4 = 0.8258 + 0.00034 * (H)
43:
44:       # Unscaled CHF result
45:       q_westing = (factor_1 * factor_2 * factor_3 * factor_4)
46:
47:       q_westing = q_westing.requires_grad_(True)
48:
49:       # Scaled CHF result
50:       W3 = scaler_y.transform(q_westing)
51:
52:       # Partial Derivatives Computation
53:       dW3_dD = torch.autograd.grad(W3, Ds,
                        grad_outputs=torch.ones_like(W3), create_graph=True)[0]
54:       dW3_dP = torch.autograd.grad(W3, Ps,
                        grad_outputs=torch.ones_like(W3), create_graph=True)[0]
55:       dW3_dG = torch.autograd.grad(W3, Gs,
                        grad_outputs=torch.ones_like(W3), create_graph=True)[0]
56:       dW3_dX = torch.autograd.grad(W3, Xs,
                        grad_outputs=torch.ones_like(W3), create_graph=True)[0]
57:
58:       return dW3_dD, dW3_dP, dW3_dG, dW3_dX
59:
60:
61:    def loss_physics (model_CHF, pd_corr, D, P, G, X) :
62:
63:       # DEFINITION INPUTS
64:       # 1.   model_CHF is the scaled neural network prediction of CHF
65:       # 2.   pd_corr is the set of partial derivatives of the correlation
66:       # 3.   D, P, G, X are some of the scaled tensors used in computing
          #      model_CHF
67:       #
68:       # NOTES FOR USE
69:       # - variable L is missing from the matrix used for computing model_CHF
          #   because such variable is not used for the correlation and thus no pd can
          #   be calculated
70:
71:       dW3_dD, dW3_dP, dW3_dG, dW3_dX = pd_corr
72:
73:       # Partial derivative model with respect to each variable
74:       dmodel_dD = torch.autograd.grad(model_CHF, D, grad_outputs=
                           torch.ones_like(model_CHF), create_graph=True)[0]
```

```
75:         dmodel_dP = torch.autograd.grad(model_CHF, P, grad_outputs=
                        torch.ones_like(model_CHF), create_graph=True)[0]
76:         dmodel_dG = torch.autograd.grad(model_CHF, G, grad_outputs=
                        torch.ones_like(model_CHF), create_graph=True)[0]
77:         dmodel_dX = torch.autograd.grad(model_CHF, X, grad_outputs=
                        torch.ones_like(model_CHF), create_graph=True)[0]
78:
79:         # Derivative difference and Loss_Physics formulation
80:         loss_dD = (dmodel_dD - dW3_dD) ** 2
81:         loss_dP = (dmodel_dP - dW3_dP) ** 2
82:         loss_dG = (dmodel_dG - dW3_dG) ** 2
83:         loss_dX = (dmodel_dX - dW3_dX) ** 2
84:
85:         loss_physics = loss_dD + loss_dP + loss_dG + loss_dX
86:
87:         return loss_physics
88:
89:
90:     def loss_total (model_CHF, true_CHF, pd_corr, D, L, P, G, X,
                        scaler_x, lamb) :
91:
92:         # DEFINITION INPUTS
93:         # 1.   model_CHF is the scaled neural network prediction of CHF
94:         # 2.   true_CHF is the scaled tensor containing all actual values
95:         # 3.   pd_corr is the set of partial derivatives of the correlation
96:         # 4.   D, P, G, X are some of the scaled tensors used in computing
            #      model_CHF
97:         # 5:   lambda is the tuning parameter to be set in the cycle
98:
99:         # Parameters validity range for Westinghouse-3 correlation
100:        # Physics loss to be used only here
101:        scaled = torch.stack([D, L, P, G, X], dim=1)
102:        unscaled = scaler_x.inverse_transform(scaled)
103:
104:        Du = unscaled[0: , 0]
105:        Pu = unscaled[0: , 2]
106:        Gu = unscaled[0: , 3]
107:        Xu = unscaled[0: , 4]
108:
109:        D_top = 0.018
110:        D_bot = 0.005
111:
```

```
112:    P_top = 15860 #kPa
113:    P_bot = 6890  #kPa
114:
115:    G_top = 6781
116:    G_bot = 1356
117:
118:    X_top = 0.15
119:    X_bot = -0.15
120:
121:    validity_range = (
122:        (D_bot <= Du) & (Du <= D_top) &
123:        (P_bot <= Pu) & (Pu <= P_top) &
124:        (G_bot <= Gu) & (Gu <= G_top) &
125:        (X_bot <= Xu) & (Xu <= X_top)
126:    )
127:
128:    # Computation of the total_loss
129:    loss_data = (true_CHF - model_CHF) ** 2
130:    loss_total = torch.zeros_like(loss_data)
131:
132:    if lamb > 0 :
133:
134:        loss_phys = loss_physics(model_CHF, pd_corr, D, P, G, X)
135:
136:        for i in range(loss_data.size(0)) :
137:
138:            if validity_range[i].item() :
139:                loss_total[i] = (1 - lamb) * loss_data[i] + lamb * loss_phys[i]
140:            else :
141:                loss_total[i] = loss_data[i]
142:
143:    else :
144:
145:        loss_total = loss_data
146:
147:    loss = torch.mean(loss_total)
148:    # print(f'loss_total: {loss_total}')
149:
150:    return loss, torch.mean(loss_data)
```