Fall 12-2022

Artificial Neural Network for Predicting Heat Transfer Rates in Supercritical Carbon Dioxide

Vinusha Dasarla Giri Babu
Embry-Riddle Aeronautical University, DASARLAV@my.erau.edu

ARTIFICIAL NEURAL NETWORKS FOR PREDICTING HEAT TRANSFER RATES IN SUPERCRITICAL CARBON DIOXIDE

By

Vinusha Dasarla Giri Babu

A Thesis Submitted to the Faculty of Embry-Riddle Aeronautical University

In Partial Fulfillment of the Requirements for the Degree of

Master of Science in Aerospace Engineering

December 2022

Embry-Riddle Aeronautical University

Daytona Beach, Florida

ARTIFICIAL NEURAL NETWORKS FOR PREDICTING

HEAT TRANSFER RATES IN SUPERCRITICAL CARBON DIOXIDE

By

Vinusha Dasarla Giri Babu

This Thesis was prepared under the direction of the candidate's Thesis Committee Chair, Dr. Mark Ricklick, Department of Aerospace Engineering, and has been approved by the members of the Thesis Committee. It was submitted to the Office of the Senior Vice President for Academic Affairs and Provost, and was accepted in the partial fulfillment of the requirements for the Degree of Master of Science in Aerospace Engineering.

THESIS COMMITTEE

Mark Ricklick Digitally signed by Mark Ricklick Date: 2022.12.01 14:22:43 -05'00'

Chair, Dr. Mark Ricklick

Sandra Boetcher Digitally signed by Sandra Boetcher Date: 2022.12.04 12:43:38 -05'00'

Co-Chair, Dr. Sandra Boetcher

Prashant Shekhar Digitally signed by Prashant Shekhar Date: 2022.12.06 09:58:17 -05'00'

Member, Dr. Prashant Shekhar

Habib Eslami 2022-12-05 21:39-05:00

Member, Dr. Habib Eslami

Hever Moncayo Digitally signed by Hever Moncayo Date: 2022.12.06 12:10:31 -05'00'

Graduate Program Coordinator, Dr. Hever Moncayo

12/06/2022

Date

James Gregory Digitally signed by James Gregory Date: 2022.12.07 14:50:33 -05'00'

Dean of the College of Engineering, Dr. James W. Gregory

10/07/2022

Date

Christopher Grant Digitally signed by Christopher Grant Date: 2022.12.07 15:51:17 -05'00'

Associate Provost of Academic Support, Dr. Christopher Grant

12/07/2022

Date

ACKNOWLEDGMENTS

I would like to thank my research advisor and thesis committee chair Dr. Mark Ricklick for his guidance and support. I also extend my gratitude to the members of my committee for helping me complete the project. This research would not have been possible without your advice and supervision.

I am also thankful for all the support I received from friends, lab mates and the sCO_2 research group. I'm extremely grateful to my parents and sister for always believing in me and pushing me to be a better person. Lastly, I'd like to mention my niece Akira, welcome to the family and thank you for inspiring me to do more.

ABSTRACT

Supercritical carbon dioxide as a working fluid in a closed Brayton cycle is proving to be more efficient than a conventional steam-based Rankine engine. Understanding the heat transfer properties of supercritical fluids is important for the design of a working engine cycle. The thermophysical properties of supercritical fluids tend to vary non-linearly near the pseudo-critical region. Traditionally, empirical correlations are used to calculate the heat transfer coefficient. It has been shown in the literature and within our own studies that these correlations provide inaccurate predictions near the pseudo critical line, where heat transfer may be deteriorated or enhanced, resulting from strong buoyancy and acceleration effects, and strong variations in fluid properties. The current study successfully uses machine learning techniques to capture these non-linearities and complex physics, providing an accurate tool for the design of heat transfer devices. The dataset is generated using highly validated computational fluid dynamics analysis. The bulk temperature and wall temperature data was obtained for a range of heat flux (q = [6, 12, 24, 36, 48]) and mass flux (G = [200, 400, 600, 800, 1000]) conditions. An artificial neural network base model was trained, validated, and tested using the CFD data. The test case was strategically selected such that the artificial neural network model trained on the high heat flux and mass flux (extreme) cases. Using the base model, hyperparameter tuning was performed, bringing down the prediction error on the test case by 94%. The final model predicted on the test set with an error less than 1%. This approach is computationally cost efficient compared to the traditional correlation-based approach as it took only few minutes for the model to train and predict. Lastly, this study published an artificial neural network tool that can be used to predict the wall temperature. Establishing a machine learning model capable of accurately predicting the wall temperature will aid in the design and development of future power generation cycles.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

NOMENCLATURE

λ	thermal conductivity
μ	viscosity
ρ	density
C_p	specific heat
CO_2	carbon dioxide
sCO_2	supercritical carbon dioxide
T_b	bulk temperature
T_w	wall temperature
T_{pc}	pseudo-critical temperature
G	mass flux
Gr	Grashof number
h	heat transfer coefficient
k	thermal conductivity
Nu	Nusselt number
Pr	Prandtl number
q	heat flux
Re	Reynolds number

1 Introduction

1.1 Motivation

Supercritical fluids have numerous applications [4–10] as working fluids due to their superior heat transfer characteristics as compared to traditional sub critical fluids. Any fluid held above its critical pressure and temperature is termed as a supercritical fluid. They have accessible critical pressures and temperatures and so are used as solvents for different industries such as the food, textile, cosmetic, petroleum, power generation, etc. These proposals have increased as we work towards reducing our ecological impact.

There is a growing need for the development of alternative technological process with minimized environmental impact, reduced energy consumption and lesser toxic residues [11]. Although supercritical fluids are highly regarded for being a green solvent, the prediction of their performance through traditional approaches continues to be a challenge. This stems from the fact that the fluid properties vary drastically near the critical point, affecting flow physics and predictive capabilities. To fully take advantage of supercritical fluids in heat transfer applications, a novel approach is needed.

2 Literature Review

2.1 Supercritical Carbon-dioxide

2.1.1 Supercritical fluids

The phase diagram as shown in fig.2.1 represents the physical state of a substance at any given pressure and temperature. In the phase diagram, as the temperature and pressure increases, liquids become less dense and gases become more dense and at a certain point liquid phase and gaseous phase co-exist. This point where densities of liquid and gas are identical is called the critical point. The pressure and temperature corresponding to this point are referred as the critical pressure and critical temperature. When an element is subjected to a pressure and temperature above critical point, the it becomes supercritical. Supercritical fluids exhibit important characteristics, such as compressibility, homogeneity, and a continuous change from gas-like to liquid-like properties.

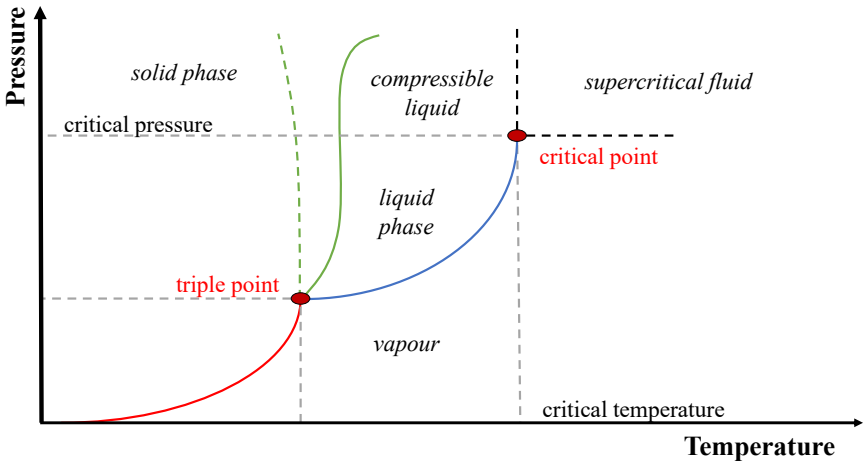

Figure 2.1 Phase Diagram of Carbon Dioxide

2.1.2 Property variations of CO_2 in the critical region

Supercritical carbon dioxide is attained by holding the fluid pressure and temperature above 7.39 MPa and 304.19 K respectively. The fluid properties of carbon dioxide vary drastically in the supercritical region and are highly sensitive to small changes in temperature and pressure. Figure 2.2 represents the variation of thermophysical properties such as

specific heat (C_p), density (ρ), thermal conductivity (λ) and viscosity (μ) at a pressure of 8MPa. From the figure, pseudo-critical temperature or T_{pc} is defined as the temperature at which specific heat reaches a maximum. For a fixed pressure, the density and viscosity decrease sharply near the pseudo-critical temperature. The peaks of specific heat and thermal conductivity are in the vicinity of the pseudo-critical temperature.

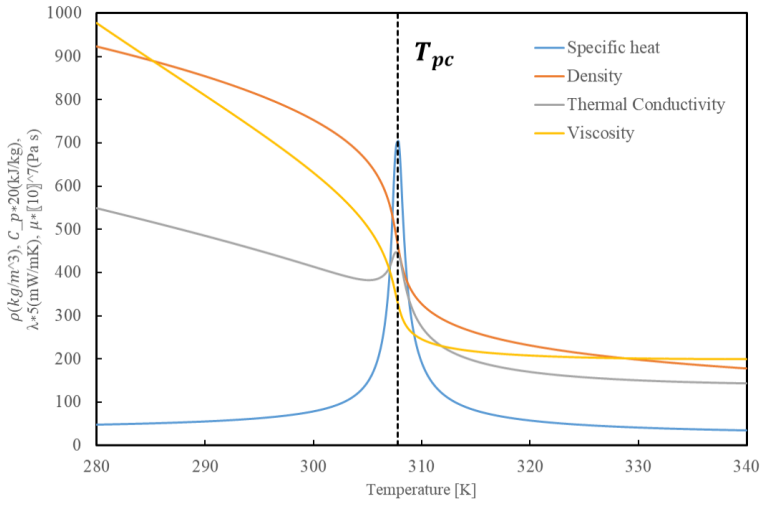

Figure 2.2 Thermo-Physical properties of sCO_2 at P = 8MPa [1]

Figures 2.3-2.6 shows the variation of thermal conductivity, viscosity and specific heat vs temperature plotted for pressure values ranging from 8 MPa - 12 MPa. In fig.2.3, as the temperature increases, the thermal conductivity for different pressure values decreases gradually except at 8 MPa, where there is a sudden peak in temperature. Similar trends can be observed in fig.2.5, where specific heat C_p for 8 MPa has a peak in specific heat. This spike in C_p can be associated with the fact that it tends to ∞ at critical point. Hence from the figures we can observe that the thermophysical properties vary sharply near the pseudo-critical region and for assessing the heat transfer rates in this region, we require complex heat transfer analysis, especially near the critical pressure.

2.1.3 Correlation based approach to predict the heat transfer rates of sCO_2

Since the 1950s, there has been ongoing research on the heat transfer behaviour of supercritical fluids. These studies primarily focus on using supercritical fluids as a coolant in

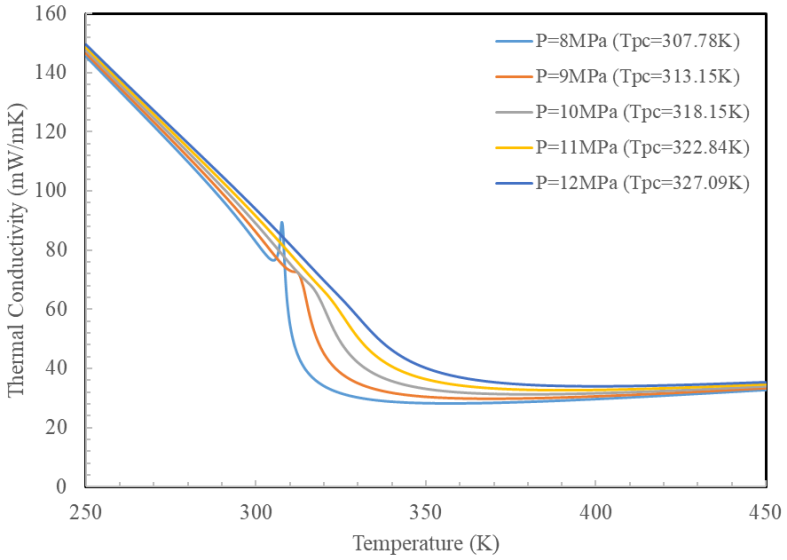

Figure 2.3 Thermal conductivity versus temperature [1]

nuclear reactors. For heat transfer analysis, dimensionless Nusselt number (Nu) is correlated against Reynolds (Re), Prandtl (Pr) and geometry. Nusselt number is the ratio of conductive resistance to the convective resistance (eq.2.1). Bringer and Smith [12] performed the experimental analysis of sCO_2 in a horizontal tube under uniform heating conditions and produced a correlation for local heat transfer (eq.2.2). Here C is a constant and is equal to 0.0375 for sCO_2. They use different reference temperatures such as bulk temperature T_b, T_{pc} pseudo critical temperate and wall temperature T_w to evaluate parameters of Reynolds number (eq.2.2).

$$Nu = \frac{hL}{k} \tag{2.1}$$

where L is the characteristic length

$$Nu = C\, Re_x^{0.77}\, Pr_w^{0.55} \tag{2.2}$$

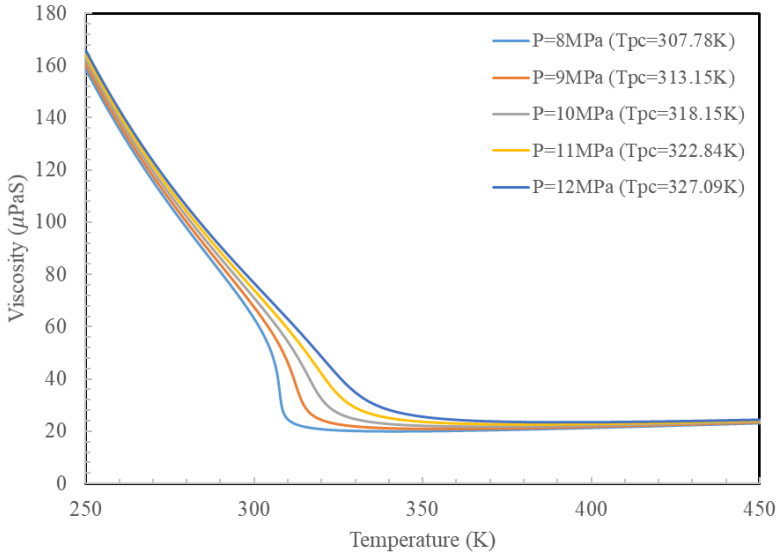

Figure 2.4 Viscosity versus temperature [1]

$$T_x = \begin{cases} T_b & \text{if } \frac{T_{pc}-T_b}{T_w-T_b} < 0 \\ T_{pc} & \text{if } 0 \leq \frac{T_{pc}-T_b}{T_w-T_b} \leq 1 \\ T_w & \text{if } \frac{T_{pc}-T_b}{T_w-T_b} > 1 \end{cases} \tag{2.3}$$

In the 1970s, experimental investigation of tube-in-tube heat exchanger under heating conditions was performed by Kransoshchekov and Protopopov [13] and after few modifications to the density correcting factor, the local heat transfer correlation of sCO2 was given by eq.2.4. Here the Nu_o which is based of the Petukhov and Kirillov [14] correlation (eq.2.6) and ϵ which is the Filonenko correlation (eq.2.7) are evaluated using wall properties. Later, Krasnoshechekov et al. [15] incorporated length factor into eq.2.8 that accounted the thermal development near the tube entrance.

$$Nu = Nu_0 \left(\frac{\bar{c}_p}{c_{p,b}}\right)^n \left(\frac{\rho_w}{\rho_b}\right)^m \tag{2.4}$$

$$m = 0.35 - 0.05 \left(\frac{P}{P_c}\right) \tag{2.5}$$

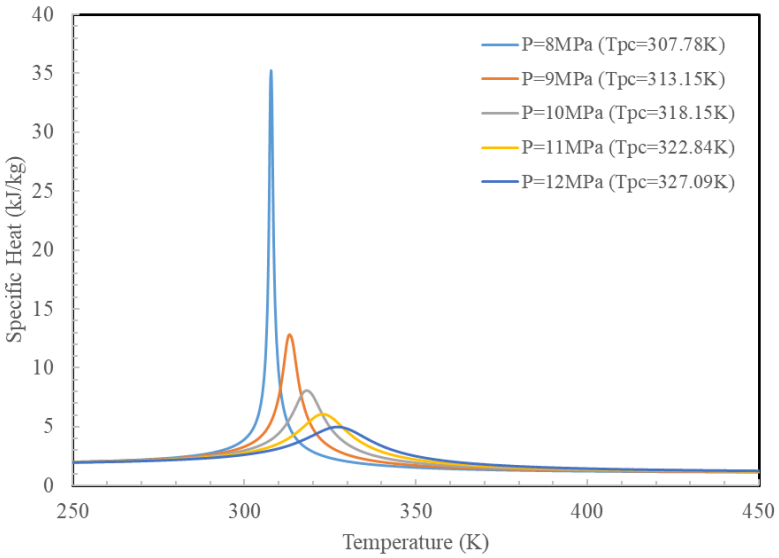

Figure 2.5 Density versus temperature [1]

$$Nu_0 = \frac{(\xi/8)\, Re\, Pr}{1.07 + 12.7\sqrt{\xi/8}\,(Pr^{2/3} - 1)} \tag{2.6}$$

$$\xi = \frac{1}{(1.82\, log_{10}(Re)\, - 1.64)^2} \tag{2.7}$$

$$Nu = Nu_0 \left(\frac{\bar{c}_p}{c_{p,b}}\right)^n \left(\frac{\rho_w}{\rho_b}\right)^{0.3} f\left(\frac{x}{d}\right) \tag{2.8}$$

where

$$f\left(\frac{x}{d}\right) = \begin{cases} 1 & \text{for } x/d > 15 \\ 0.95 + 0.95\left(\frac{d}{x}\right)^{0.8} & \text{for } 2 \leq x/d \leq 15 \end{cases} \tag{2.9}$$

The most commonly referred Gnielinski [16] correlation was obtained modifying the Petukhov et al. [17]. This correlation predicts the heat transfer rates over a transitional range Re $= 2300 - 10^4$. Equation 2.10 was further modified to account the tube length and local temperature resulting in eq.2.11.

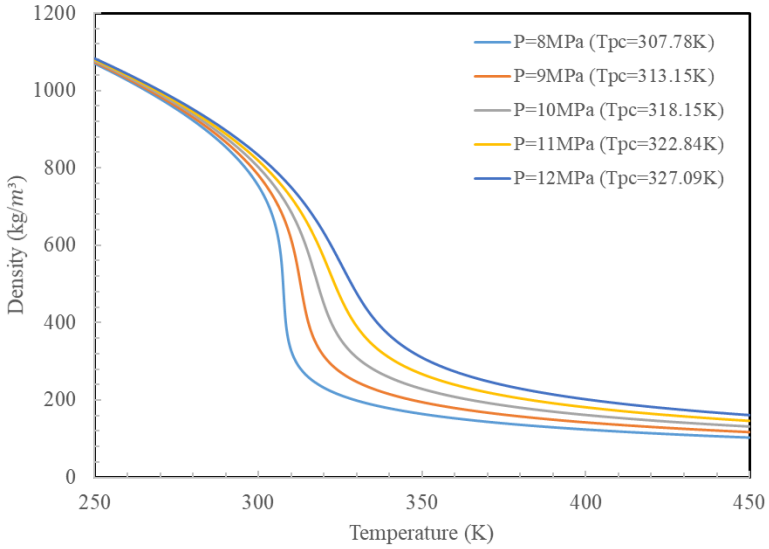

Figure 2.6 Specific heat versus temperature [1]

$$Nu = \frac{(\xi/8)\,(Re - 1000)\,Pr}{1 + 12.7\,\sqrt{\xi/8}\,(Pr^{2/3} - 1)} \tag{2.10}$$

$$Nu = \frac{(\xi/8)\,(Re - 1000)\,Pr}{1 + 12.7\,\sqrt{\xi/8}\,(Pr^{2/3} - 1)} \left[1 + \left(\frac{d}{l}\right)^{2/3}\right] K \tag{2.11}$$

where

$$K = \begin{cases} \left(\frac{Pr_b}{Pr_w}\right)^{0.11} & \text{for liquids in the range } \frac{Pr_b}{Pr_w} = 0.05 - 20 \\ \left(\frac{T_c}{T_w}\right)^{0.45} & \text{for gases in the range } \frac{T_c}{T_w} = 0.5 - 1.5 \end{cases} \tag{2.12}$$

In the early 2002s, Pitla et al. [18] performed numerical investigation, experimental analysis of supercritical carbon dioxide in a series of horizontal tubes and developed the correlation (eq.2.13) where Nusselts number is evaluated using bulk and wall properties.

$$Nu = \left(\frac{Nu_w + Nu_b}{2}\right) \frac{k_w}{k_b} \tag{2.13}$$

where Nu_w and Nu_b are evaluated using

$$Nu = \frac{(\xi/8)\,(Re - 1000)\,Pr}{1.07 + 12.7\,\sqrt{\xi/8}\,(Pr^{2/3} - 1)} \tag{2.14}$$

Liao et al. [19] performed experimental analysis of supercritical carbon dioxide in horizontal tubes of various diameters under cooling conditions (eq.2.15). They also developed correlations for horizontal flow, upward and downward flow (eq.2.17-eq.2.19). Here, the buoyancy effects on are accounted including Ri_b and Bu parameters.

$$\frac{Nu}{Nu_{db}} = 5.57 Ri_b^{0.205} \left(\frac{\rho_b}{\rho_w}\right)^{0.437} \left(\frac{\overline{c}_p}{c_{p,w}}\right)^{0.411} \tag{2.15}$$

where

$$Nu_{db} = 0.023\,Re^{0.8}\,Pr^{0.3} \tag{2.16}$$

is the Dittus-Boelter correlation for cooling

For horizontal flow

$$Nu = 0.124\,Re_b^{0.8}\,Pr_b^{0.4} Ri_b^{0.203} \left(\frac{\rho_w}{\rho_b}\right)^{0.842} \left(\frac{\overline{c}_p}{c_{p,b}}\right)^{0.384} \tag{2.17}$$

For upward flow

$$Nu = 0.354\,Re_b^{0.8}\,Pr_b^{0.4}\,Bu^{0.157} \left(\frac{\rho_w}{\rho_b}\right)^{1.297} \left(\frac{\overline{c}_p}{c_{p,b}}\right)^{0.296} \tag{2.18}$$

For downward flow

$$Nu = 0.643\,Re_b^{0.8}\,Pr_b^{0.4}\,Bu^{0.186} \left(\frac{\rho_w}{\rho_b}\right)^{2.154} \left(\frac{\overline{c}_p}{c_{p,b}}\right)^{0.751} \tag{2.19}$$

where Bu is the buoyancy parameter defined by eq.2.20 from Jackson and Hall [20].

$$Bu = \frac{\overline{Gr}_b}{Re_b^{2.7}} \tag{2.20}$$

In 2004, Dang and Hihara [2] investigated sCO_2 the heat transfer rates of supercritical carbon-dioxide in a horizontal tube. A new correlation (eq.2.21) was proposed by modifying the Gnielinski correlation (eq.2.11) and basing it on the bulk, wall and film temperature.

$$Nu = \frac{(\xi_f/8)\,(Re_b - 1000)\,Pr}{1.07 + 12.7\,\sqrt{\xi_f/8}\,(Pr^{2/3} - 1)} \tag{2.21}$$

where

$$Pr = \begin{cases} \frac{c_{p,b}\,\mu_b}{k_b} & \text{for } c_{p,b} \geq \overline{c}_p \\ \frac{\overline{c}_p\,\mu_b}{k_b} & \text{for } c_{p,b} < \overline{c}_p \text{ and } \frac{\mu_b}{k_b} \geq \frac{\mu_f}{k_f} \\ \frac{\overline{c}_p\,\mu_f}{k_f} & \text{for } c_{p,b} < \overline{c}_p \text{ and } \frac{\mu_b}{k_b} < \frac{\mu_f}{k_f} \end{cases} \tag{2.22}$$

This section covers a few of the main correlations and the operating conditions for all these correlations are provided in Appendix (Table A.1). An extensive review of all the existing correlations for supercritical carbon dioxide can be found in the review paper by Lopes et al.[21]. The paper covers all the modifications of the existing correlations and identifies the errors made. The addition of correcting factors, buoyancy and flow acceleration combined with the advancement of thermal and flow properties has made heat transfer correlations complex. This has resulted in the the development of multiple correlations. Apart from developing a correlation for sCO_2 in horizontal tube by performing experimental analysis, Dang et al.[2] also compares few of the existing correlations against experimental data. Figure 2.7 is taken from the paper, it shows different correlations Gnielinski (eq.2.11), Petro-Popov [22], Pitla (eq.2.13), Liao (eq.2.15) and Yoon [23] model compared against the experimental data (measured data). Most of these correlations were discussed in the previous section. The operating conditions are heat flux q = 24 kW/m^2, mass flux G = 200 kg/m^2s, diameter = 6mm, pressure = 8 MPa. This is considered as high heat flux and low mass flux case, here the distribution of properties in the radial direction has considerable effect on the heat transfer coefficient.

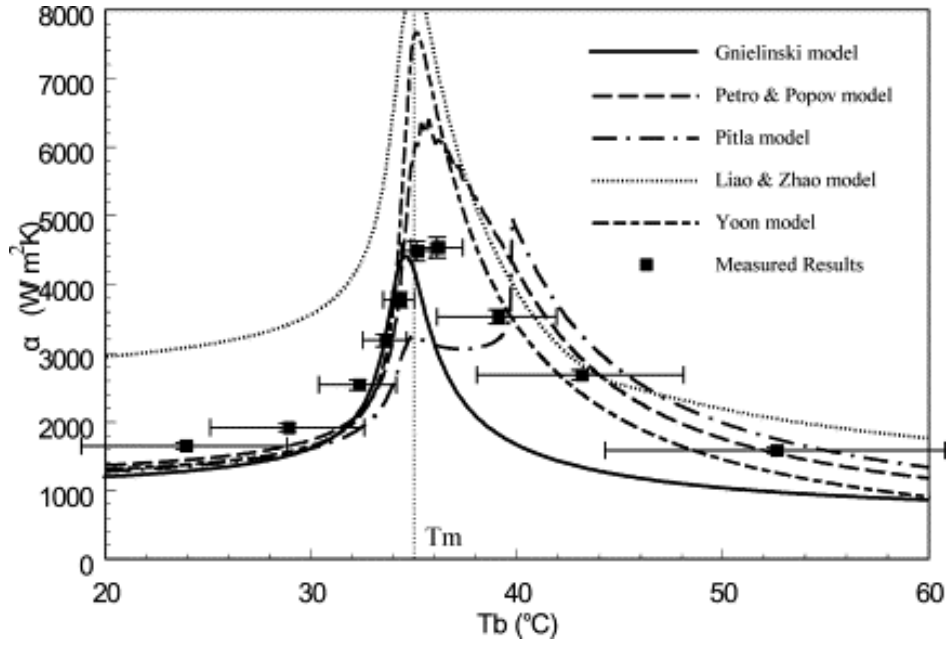

Figure 2.7 Comparison of heat transfer coefficient α calculated using various correlations and measured α [2]

In the fig. 2.7, the heat transfer coefficients (α) are plotted against the bulk temperature (T_b). Note that since all the fluid properties in a tube depend on the bulk temperature, it would be efficient to represent data in terms of bulk temperature. The T_m line in the figure is the pseudo critical temperature. When $T_b < T_m$, the Gnielinski correlation predicts well relative to the experimental data. But when $T_b > T_m$, the correlation under-predicts by about 30%. The Petro-Popov model performs well in the regions $T_b < T_m$ and $T_b > T_m$ but overestimates the peak by 30%. Similarly the Yoon model over predicts the peak by 50%. Dang et al. [2] states that the reason Pitla model why doesn't do well is because the model was based on a different pressure range (P= 9.4MPa to 13.4MPa) and a large mass flux (G = 1152 to 2300 kg/m^2). Since is there is discontinuity in the operating conditions, the model fails after T_m. The Liao model is stated to be highly sensitive to heat flux and so it overestimates the heat transfer coefficient at high heat flux.

In 2011, Fang et al. [3] conducts a comprehensive review on the available experimental data and correlations for in-tube heat transfer of sCO_2 under cooling conditions. The paper compares the 12 correlations(Krasnoshchekov[13], Baskov[24], Petrov-Popov[22, 25], Fang[3],

Liao[19], Pitla[18], Yoon[23], Dang[2], Huai[26, 27], Son-Park[28], Kuang[29], Oh-Son[30]) with experimental data from Dang et al. [2]. The fig.2.8 is pulled from the paper. The heat transfer coefficient α is compared against bulk temperature over pseudo-critical temperature ($\frac{T_b}{T_{p,c}}$) for q = 24 kW/m^2, mass flux G = 200 kg/m^2s, diameter = 6mm, pressure = 8 MPa. Note that these operating conditions are similar to the conditions discussed in the previous paper but the the x-axis label is represented in a different manner. It was reported that the Petrov-Popov [25] and Fang [3] correlations performed the best. We can observe huge peaks produced from the Oh-Son, Yoon and Son-Park model. The correlations of Kuang, Huai, and Pitla models can not predict the trends after the pseudo-critical points if heat flux or mass flux is greater. Although several correlations are available, there has not been a significant increase in the prediction capabilities [31, 32]. From literature review we observe that there are several correlations for one operating condition. Most of these correlations under or over-predict near the pesudocritical region. Empirical based approach is not able to model the complex heat transfer behaviour of supercritical carbon dioxide. And so, researchers have started to explore other methods to predict the heat transfer rates of supercritical carbon dioxide.

2.2 Machine Learning

The idea of Machine Learning was first proposed by Alan Turing in the 1950s [33]. His paper posed the question "Can machines think?". In the late 1950s, IBM computer scientist Arthur Samuel wrote a computer program to play checkers. The performance of the computer program was improved by playing thousands of games against itself and by mid 1970s, the program was capable of playing as efficiently as a amateur human player. But the foundation for machine learning started way back in the 18th and 19th centuries where the fundamentals of machine learning concepts such as Bayes Theorem and the method of Least Squares were introduced [34]. Machine Learning is generally defined as the development and usage of computer systems that uses algorithms and statistical models to analyze and draw inferences from patterns in data. These systems must be capable of learning without any

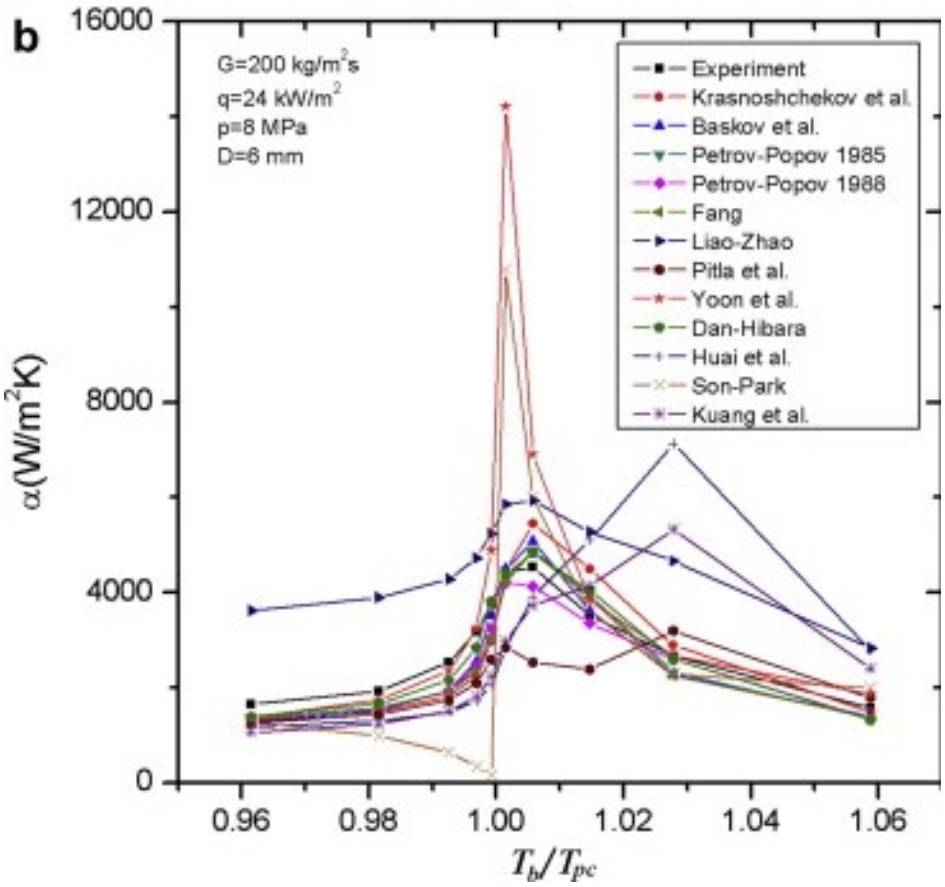

Figure 2.8 Predictions of existing models vs. experimental data [3]

explicit instructions. While there are several interpretations of what machine learning means, the universally accepted way to describe machine learning is given by Tom M. Mitchell. He defines machine learning as " A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E ". This statement implies that a machine is considered to be learning if it gains experience by doing a task and improves it's efficiency by performing the similar tasks in the future.

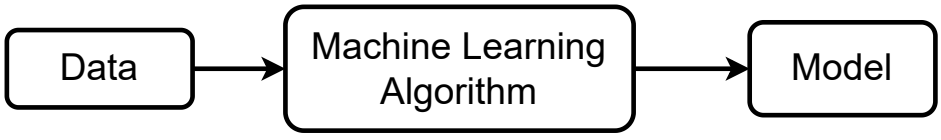

Figure 2.9 Components of a Machine Learning Model

The main components of a machine learning model is shown in fig.2.9. The data usually contains patterns. The machine learning algorithms recognizes and learns these patterns. Once the algorithm has completed the learning process, a machine learning model is generated. This trained model should be capable of identifying the patterns when new data is fed to the model. Before discussing the different types of machine learning, it is important that we define a well posed machine learning problem. It requires a specific task, performance metrics and source of training experience. For example, the classifying spam emails is a well posed machine learning problem. The task is to identify spam emails, performance can be measured by checking the fraction of emails accurately classified as spam or not and the training experience is observing if we label the email as spam or not.

2.2.1 Types of Machine Learning Algorithms

Machine Learning is broadly classified into three categories: supervised learning, unsupervised learning and reinforcement learning. The most frequently used learning paradigm is supervised learning. It is task oriented and requires labeled data as input. On the other hand unsupervised learning is data driven and uses unlabeled data for training. Reinforcement learning is used when data is time dependent or has hysteresis.

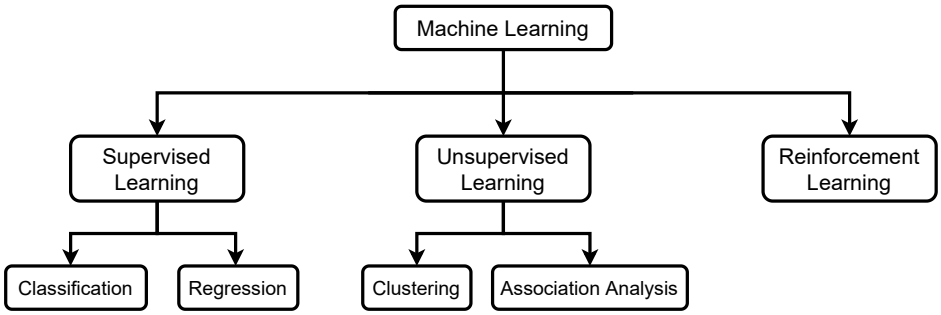

Figure 2.10 Types of Machine Learning

Supervised Learning

Supervised Machine learning is based on learning information from the past. Referring to the definition of a well posed machine learning problem, for supervised learning the past information is the training experience. This can be further explained by considering the

following example. If a parent wants the child to learn the colours of an object, the parent teaches the child using basic information. Supervised learning also requires fundamental information which can be provided in the form of 'training data'. This training data is tagged with 'labels'. For a computer to identify different colours of objects, the training data will contain images of objects with their respective colours as their labels. Supervised learning can be further segregated into classification and regression problems. Predicting a categorical or nominal variable is a classification problem. In this case, the labelled training data is fed to a classifier algorithm. Once the trained model is obtained, it should be able to classify new data. This new data on which the trained model predicts is termed as the 'test data set'. For a regression problem, a continuous variable is predicted based on the value of one or more predictor variables. The input data/ training data should must feature variables and target variables. Features are the data elements that are analysed and targets are labels true values on which prediction is made. Consider this example, fig.2.11 shows a data snippet obtained from a public data set. The iris data set contains the following attributes, sepal length, sepal width, petal length and petal width. These form the features of this data set. Based on these features, a prediction is made on the species of the iris flower. The species columns contains labels or target values for this data set.

index	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
147	6.5	3.0	5.2	2.0	2
32	5.2	4.1	1.5	0.1	0
141	6.9	3.1	5.1	2.3	2
22	4.6	3.6	1.0	0.2	0
81	5.5	2.4	3.7	1.0	1
33	5.5	4.2	1.4	0.2	0
15	5.7	4.4	1.5	0.4	0
61	5.9	3.0	4.2	1.5	1
95	5.7	3.0	4.2	1.2	1
34	4.9	3.1	1.5	0.2	0

Figure 2.11 Iris data set with feature variables and target variables

The classification and regression problems can be solved using different algorithms [35]. The most common ones are listed below

- Classification - Naive Bayes, Logistic Regression, K-Nearest Neighbours, Support Vec-

tor Machines(SVM), Random Forests, Decision Trees and Rule based Classifiers .

- Regression - Linear, Polynomial, Lasso and Ridge regression.

Unsupervised Learning

In Unsupervised learning there is no labelled data. Here the goal is get input data and sort the data into different groups based on the patterns in the data elements. Unsupervised learning is data driven as in there aren't any specific tasks assigned. The data given as input drives the algorithm to group or organize similar data together. Clustering is one of main type of unsupervised learning. Here data is grouped together by applying a measure of similarity. The most commonly used measure of similarity is distance. The data elements are considered to be in the same cluster if the distance between them is less. Another form of unsupervised learning is association analysis, where the the relationship/association between the data is identified.

Reinforcement Learning

In this types of learning, the model learns by trial and error. Unlike the other learning paradigms, reinforcement learning is centered on interacting with the environment. The model improves its efficiency by performing the task and if it is successful, the model is rewarded. This learning process is autonomous and model learns from its mistakes.

Table 2.1 Practical Applications of the different Learning Paradigms

Supervised	Unsupervised	Reinforcement
Handwriting recognition	Market based analysis	Self driving cars
Stock market prediction	Customer segregation	intelligent robots
Fraud detection	User recommendation	AI games

2.2.2 Applications of Machine Learning

Machine Learning is becoming increasingly popular and is being used for various applications because of its intelligent learning capabilities [35]. Banking and finance industries use

machine learning to identify fraudulent transactions on a real time basis. Predictive learning can be used to identify customers who are more likely to switch banks [36]. Risk management is essential for insurance industries, machine learning can be used to asses risk of a potential customer [37]. Disease control and management is an important sector under the Healthcare industry. Machine learning algorithms can be used to forecast where or when the disease is likely to spread. Ahmad et al [38] reviews the application of interpretable machine learning in healthcare. Machine learning is becoming increasing popular in autonomous vehicle industry, smart electric power systems, image processing in engineering fields, renewable energy and heat transfer. Kwon et al. [39] reviews the application of random forest algorithm to predict the convection heat transfer coefficients for a high-order nonlinear heat transfer problem. Artificial neural networks are used by Baghban et al. [40] to predict the heat transfer rates of a helically coiled tube. In a attempt to develop a condensation heat transfer coefficient for a cooling system, Lee et al. [41] used convolution neural network approach for the study. Alizadeh et al. [42] used artificial intelligence based physics models to predict the transport and thermodynamic processes in multi-physics systems. Recently, a lot of research is based on implementing artificial neural networks to predict the heat transfer coefficients.

2.3 Artificial Neural Networks

2.3.1 Brief historical review

Artificial Neural Networks, commonly used to solve complex machine learning problems, are capable of identifying non linear and dynamic relationships in data. They are loosely based on biological neural networks and are made up several mathematical neurons. This idea dates back to the 1940s when McCulloch and Pitts [43] proposed their seminal work. They considered the neuron as functional logic device. This led to the proposal of a 'Perceptron Model' by Rosenblatt [44] in 1957, shown in fig.2.12. Later Minsky and Papert [45] showed the limitations of a single perceptron and this prompted a down fall in the development of the neural networks. In the 1980s, interest in artificial neural networks started to grow again when Werbos [46] developed the back propagation learning algorithm for multi-layer

perceptron. This algorithm was rebuilt several times and was well-familiarized by McClelland [47]. The detailed history of the development of Artificial Neural Networks is given by We et al. [48]

2.3.2 Artificial Neurons

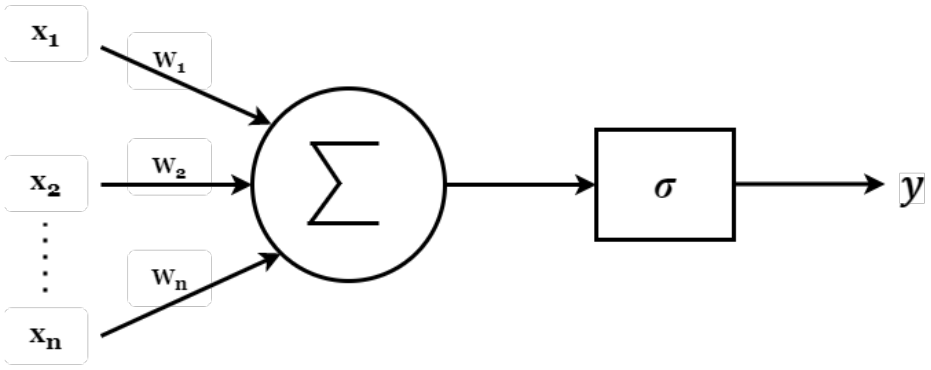

Figure 2.12 Schematic of a Perceptron

The perceptron as depicted in fig.2.12 is a computational model of an biological neuron. This mathematical neurons receives n input parameters $x_i(i = 1, 2, ...n)$. The main components of an artificial neuron are the weighted inputs, summation and activation function. The artificial neuron uses these mathematical functions to processes the inputs. The inputs are assigned weights which signifies the importance of the inputs. If the weight is a positive values, it means that that input has a positive influence on the output parameter. A negative weight would have an inhibitory effect on the output. The summation functions adds all the weighted inputs. A bias (b) is added to this summation which adjusts the input to the activation function. The last component of the artificial neural network is the activation function (σ) which produces an output ($\sigma(z)$) only when the input it receives passes a certain threshold. The activation function introduces a non linearity and defines the prediction accuracy of the neural network. In the absence of an activation function, the artificial neural network is reduced to a simple linear regression model. The output (z) of the artificial neuron is given by eq.2.23.

$$z = \left[\sum_{i=1}^{n}(x_i * w_i) + b\right] \tag{2.23}$$

Types of Activation Functions

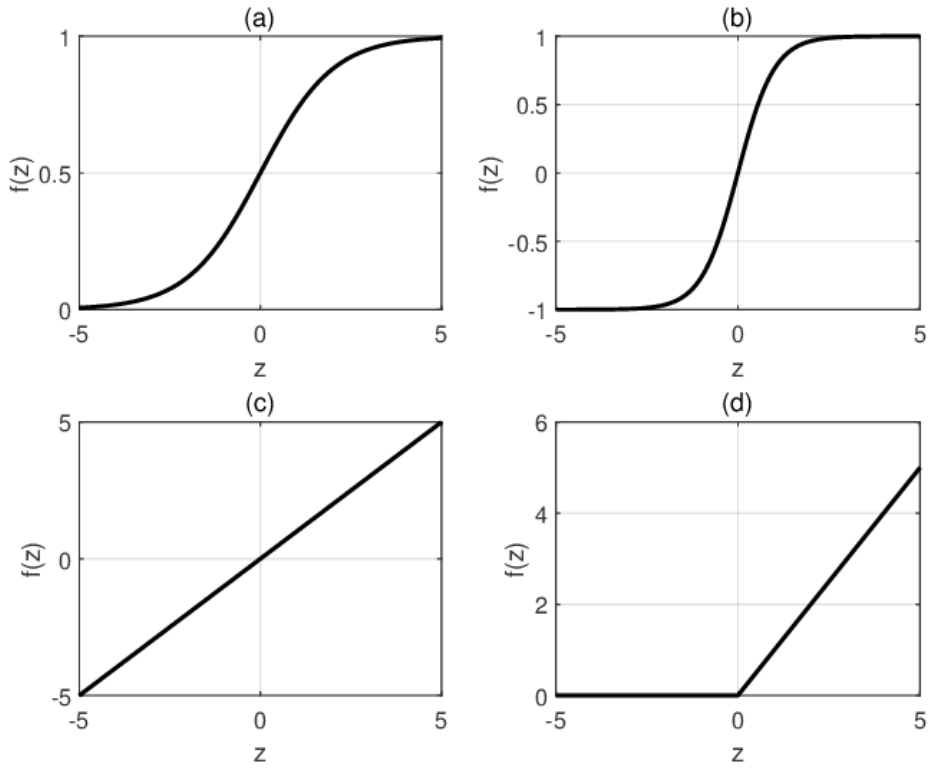

Figure 2.13 Activation Functions (a)Sigmoid Function (b)Hyperbolic Tangent Function (c)Linear Function (d)ReLU Function

The different types of activation functions are reviewed by Sharma et al. [49]. The most common ones, as shown in fig.2.13, are discussed below. The identify function is a linear function of form eq.2.24 and is usually used in the input layer. The binary step function, mathematically given by eq.2.25 produces 1 as output if the input is either positive or zero. The step function gives output as zero if the input is negative. The threshold function (eq.2.26) is similar to the step function, but instead of zero the input x is dependent on a threshold value (θ). ReLU or rectified linear unit is the most popularly used activation function. It reforms the input value between the maximum of zero and input value itself. Sigmoid activation function modifies the output values in the range of 0 and 1. This activa-

tion function is given by eq.2.29. The hyperbolic tangent function is similar to the sigmoid function. Here the activation function is symmetric at the origin.

$$y_{out} = \sigma(x),\ for\ all\ x \tag{2.24}$$

$$y_{out} = \sigma(z) = \begin{cases} 1\ if\ x >= 0 \\ 0\ if\ x < 0 \end{cases} \tag{2.25}$$

$$y_{out} = \sigma(z) = \begin{cases} 1\ if\ x >= \theta \\ 0\ if\ x < \theta \end{cases} \tag{2.26}$$

$$y_{out} = \sigma(z) = max(0, x) \tag{2.27}$$

$$y_{out} = sigma(z) = \frac{1}{1 + e^{-}z} \tag{2.28}$$

$$y_{out} = sigma(z) = \frac{e^{z} - e^{-}z}{e^{z} + e^{-}z} \tag{2.29}$$

2.3.3 Architecture of an Artificial Neural Network

The artificial neural network which is made up several interconnected neurons, can be classified into two types based on the connection pattern, feed forward neural network and recurrent neural network. Feed forward neural network is the most common unidirectional network, that is there are no loops in the network. It is commonly referred as a multi-layer perceptron model. The recurrent neural network on the other hand has feedback connections and are not unidirectional. Jain et al [50] reviews different architectures of neural networks and their respective applications. In this study, we focus on the working a feed forward neural network. The schematic of a simple feed forward neural network is shown in fig.2.14. The training or learning process of a neural network involves reforming the network and weights

such that the neural network predicts with least errors. It consists of two parts, learning paradigm and back propagation. As discussed earlier, for supervised learning we use labelled data. That is we provide the network with correct output for all the input values. In this type of learning paradigm, the weights are randomly assigned to the inputs, information is propagated through the neurons and the initial predicted output $\hat{y}$ is obtained. Since the neural network has the true values, the error between the true values (y') and predicted output is obtained. Cost function (eq.2.30) measures the loss of the entire network. Mean Squared loss or the root mean squared loss is commonly used as the loss criterion to asses regression problems.

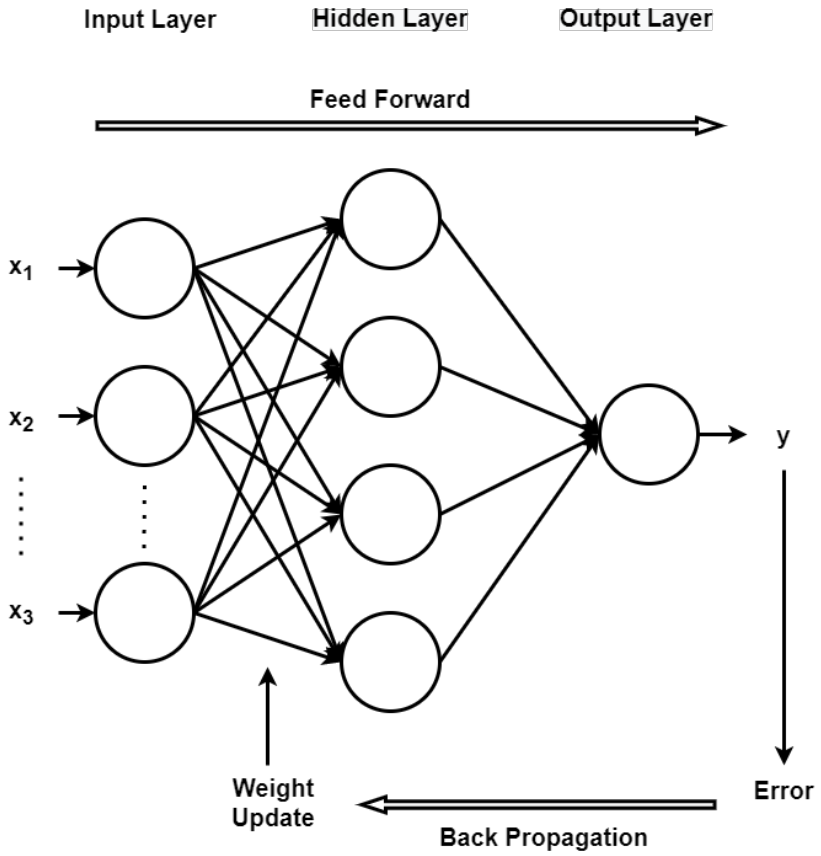

Figure 2.14 Feed Forward Neural Network

$$CostFunction(J) = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2 \tag{2.30}$$

The aim of the learning process is to reduce the cost function. This brings us to the second part of the learning process, back propagation. Based on the error obtain, the weights are adjusted and this information is propagated back into the network. This process is done using an optimization function. Gradient descent is one of the most popular algorithms used for optimization. It iteratively calculates the the local minimum of the cost function by using the negative gradient of the initial position. The gradient is scaled by a constant called the learning rate. This is termed as hyper parameter that affects the efficiency of the learning process. The Cost function (J) is a function of weights and bias, they are adjusted using gradient descent such that the cost is brought to a minimum. Equation 2.31 and 2.32 gives formulas for the adjusted weights and bias respectively. This entire process is repeated until the error reaches zero or close to zero. Artificial neural networks teach themselves the patterns in the data through the learning process. This adaptive learning technique allows the neural network to be more efficient than convectional modelling approaches.

$$w = w - learningrate * \frac{\partial J}{\partial w} \tag{2.31}$$

$$b = b - learningrate * \frac{\partial J}{\partial b} \tag{2.32}$$

The learning process of a neural network is dependent on three major aspects. 1. The number of hidden layers - there can be multiple hidden layers. As the number of layers increase, the processing time also increases. If time is an important factor, lesser number of hidden layers are preferred. If accuracy is more important factor, a deeper network is suggested i.e higher number of layers. 2. The number of neurons - this number can be varied to obtain the optimal number using train and analysis method. A larger number of nodes can led to either increased performance or over-fitting. 3. Weights - the weights assigned to a neuron plays a significant role in determining the importance of that neuron. They can corrected by varying hyperparameters such as changing the number of layers, number

of neurons, optimizer, etc.

2.3.4 Applications of Artificial Neural Networks

Artificial Neural networks has been applied to different industries. Abiodun et al. [51] summarizes the various application of artificial neural networks. They are applied to problems which has highly complex data sets. Artificial Neural Network implementation in energy systems is discussed in detail by Kalogirou et al. [52]. A relatively older industry application is clinical medicine. Baxt et al. [53] discusses the application of artificial neural networks to predict diagnosis and outcomes. Himmel et al. [54] describes the practical uses, advantages and disadvantages of artificial neural networks for chemical engineering applications. Other industrial applications include using ANNs to predict catalysis [55], quality control of foods [56, 57] and structural mechanics [58]. Scalabrin et al. [59] was the first to implement artificial neural networks to model the forced convection heat transfer for supercritical carbon dioxide inside a heated tube. The paper developed four multi-layer feed forward neural network with one hidden layer as a function of working conditions. The first architecture uses conventional dimensionless number Re, Pr and Ec as input parameters. The scaled Nusselt number is consider to be the output parameter. The optimal number of neuron in the hidden layer was identified to be 7 using trial and error analysis. The second architecture has scaled heat transfer coefficient α as the output parameter and in the input layer we have reduced pressure (P_r), reduced temperature (T_r), mass flow rate (m) and heat flux (q). The third architecture inputs was based the elements of the Krashnoschekov-Protopopov-Petukhov-Gnielinski (KPPG) correlation (eq.2.33 - eq.2.36) and the output was set to Nusselts number. This architecture was trained using the data generated by the KPPG correlation. Although 8000 data points was generated, only a fifth of those values was used for training. Lastly the fourth architecture considered independent variables reduced pressure (P_r), reduced temperature (T_r), mass flow rate (m) and the fraction of wall temperature over bulk temperature $\frac{T_w}{T_b}$. The output parameter was taken to be heat transfer coefficient α. Two versions of each of these architectures, one with a small data set and other with

enlarged data set was trained with the experimental data from Olson et al. [60]. The paper compares the average absolute deviation (AAD) of all the architectures against the KPPG correlation and concludes that the third architecture produced an AAD of 3.98% against 4.09% for the conventional equation and the fourth architecture an AAD of 2.67% against 4.30% for the conventional equation. The paper shows that there isn't evident advantage in using the dimensionless number over physical variables. It also highlights the importance of regularly distributed data which is needed to generalize the transfer equation.

$$Nu_{CP} = \frac{(\xi/8)\,(Re - 1000)\,Pr}{1 + 12.7\sqrt{\xi/8}\,(Pr^{2/3} - 1)}\left[1 + \left(\frac{d}{l}\right)^{2/3}\right] \tag{2.33}$$

$$Nu_{SC} = Nu_{CP}\left(\frac{\overline{c}_p}{c_{p,b}}\right)^{n}\left(\frac{\rho_w}{\rho_b}\right)^{0.4} \tag{2.34}$$

$$\overline{c}_p = \frac{h_w - h_b}{T_w - T_b} \tag{2.35}$$

$$n = \begin{cases} 0.4 & \text{when } \frac{\overline{c}_p}{c_{p,b}} < 1,\ \frac{T_w}{T_{pc}} < 1\text{ , and } \frac{T_b}{T_{pc}} \geq 1.2 \\ n_1 = 0.22 + 0.18\frac{T_w}{T_{pc}} & \text{when } \frac{\overline{c}_p}{c_{p,b}} < 1 \text{ and } 1 \leq \frac{T_w}{T_{pc}} < 2.5 \\ n_1 - (5n_1 - 2)\left(\frac{T_b}{T_{pc}} - 1\right) & \text{when } \frac{\overline{c}_p}{c_{p,b}} < 1 \text{ and } 1 \leq \frac{T_b}{T_{pc}} < 1.2 \\ 0.7 & \text{when } \frac{\overline{c}_p}{c_{p,b}} > 1 \end{cases} \tag{2.36}$$

Chen et al. [61] used a similar approach where experimental data from Olsen et al. [60] and KPPG correlation was compared against a Modified Radial Bias Function Network (MRBFN). These networks have universal approximation, fast learning and use Gaussian functions as threshold functions [62]. Four different combinations of input parameters and output parameter was used in this study(given in Table A.2). Out of the 1115 data points from Olson et al. [60] only 250 random points was selected to train the model. The paper compared their results with the results obtained against a simple back propagation network,

a normal RBFN and concluded that the MRBFN with 10 neurons performed better than the correlation as it was able to predict large changes in the near critical region. Pesteei et al. [63] used experimental data from Jiang et al. [64] who obtained the wall temperature of supercritical carbon dioxide in a vertical set up(upward flow). Polynomial neural networks was implemented by Pesteei et al. [63], here each layer contains units (similar to neurons) that are considered as polynomials. The mass flux (G), Reynolds number (Re), buoyancy number (Bo), axial coordinate (x) and heat flux on the inner surface (q_w) are considered to be the input parameters and heat transfer coefficient (h) is the output parameter. The root mean squared error was found to be 25.643 W/m^2K with an R^2 errors of 0.984. The model developed is concluded to be in agreement with the experimental data.

Recently, Chu et al. [65] in 2018 incorporated Direct Numerical Simulation (DNS) with Deep Neural Network (DNN). The data (2100 data points) for training the neural network was generated using DNS for 35 different operating conditions. The proposed model has 2 hidden layers with 50 neurons in each layer. The pipe diameter, inlet pressure, heat flux, inlet temperature and bulk specific enthalpy are taken as the input parameters and the wall temperature and wall shear stress are considered to be the output parameters. This paper separates their data points into only training and validation. The resulting mean percentage error on wall temperature and wall shear stress are 0.07 and 1.02 respectively. Chu et al. [65] concludes by stating that the combination of DNS and DNN model was able to produce the same accuracy as true DNS model but with lesser computational cost. Ye et al. [66] developed a artificial neural network with 5 inputs - heat flux, mass flux, tube diameter, pressure and bulk specific enthaply and wall temperature as output. They use experimental data from several papers [67–74] that obtained the heat transfer coefficient of supercritical carbon dioxide in a vertical set up. From these experimental papers, data corresponding to significant buoyancy force and acceleration force are ignored because during the occurrence of Deteriorated Heat transfer (DHT) and Increased Heat transfer (IHT) the buoyancy force and acceleration force have significant influence on the heat transfer rates. So a total of 4354

data points were obtained for 14 different operating conditions. The ANN has one hidden layer and the number of neurons in the hidden layer are varied to find the best fit. They evaluated the ANN model against different available correlations and concluded that the ANN model has the lowest standard deviation of 0.99%.

In 2021, Zhu et al. [75] performed experimental analysis to obtain the heat transfer rates of supercritical carbon dioxide in a vertical set up. The paper uses their own experimental data as well as data from literature [68, 74, 76–78]. A total of 2674 data points are used to train, validate and test the model. The input parameters and output parameters are same as Ye et al. [66] but ANN model in Zhu et al. [75] has two hidden layers. The number of neurons in the hidden layers are varied and the best fit is found by comparing the training and validation mean squared errors. The paper states that increasing the number of layers increases the model's efficiency to capture peaks during Deteriorated Heat transfer. Zhu et al. [75] compared the neural network model with correlations [79–81] & experimental data and found that the artificial neural network model has a root mean square error of 7.29%. Note that some of these correlations were developed for supercritical water. Sun et al. [82] applied artificial neural networks to predict the thermal characteristics of in-tube upward supercritical carbon dioxide flow. Genetic algorithm and back propagation was used in this study and two networks with one and two hidden layers was compared. This study used data points from several experimental papers [67, 69–73, 83–86]. The trained model is compared with the correlations and in all cases ANN performed the best with a mean relative error less than 2.8 %.

2.4 Research Gap

Throughout the years, the parameters required to train an artificial neural network model has changed slowly. Dimensionless parameters are preferred while performing experimental analysis as they reduce the amount of data required to develop correlations. But the literature review has shown that physical values and dimensionless quantities have the same effect on artificial neural networks for supercritical carbon dioxide. We observed that in the recent

years artificial neural networks use wall temperature as output parameter instead of heat transfer coefficient (α or h). Similarly diameter, pressure, heat flux, mass flux and bulk enthalpy are considered to be the input parameters when initially Re, Pr and Ec (dimensionless numbers) was used in an effort to eliminate sensitivity to reference temperature. Machine learning based application reduced the need for a large number of material properties to predict the heat transfer rates.

From literature review, usage of artificial neural networks pose as a viable solution as they capture the non linear behaviour of fluid properties in the near-critical region. But the artificial neural network application has been limited to the prediction of heat transfer rates of upward flow of supercritical carbon dioxide. There are not many artificial neural network papers that discuss the heat transfer rates of transfer supercritical carbon dioxide in a horizontal set up. Also note that the ANN papers discussed in the literature review very have less data points to train on. More data points are required to reduce the errors of a distribution which might be difficult with experimental analysis of supercritical carbon dioxide.

2.4.1 Objective

The aim of this study is to establish an artificial neural network model that has trained on a large and uniform data set and is capable of accurately predicting the heat transfer rates of supercritical carbon dioxide in a horizontal set up under cooling conditions. The data set will be generated using highly validated computational fluid dynamic analysis. The study also focuses on establishing a procedure for hyperparameter tuning of a neural network model. Lastly, an artificial network tool is published in Github that can be used to predict the wall temperature for a fixed pressure at any bulk temperature, heat flux and mass flux.

3 Methodology

This section focuses on developing and training an artificial neural network model from scratch using PyTorch Framework. PyTorch is a popular tool used in the machine learning research community. It was developed focusing on both usability and speed [87]. Google colab is a web based Jupyter Notebook that is user friendly and accessible by everyone. Colab is capable of accessing all the data science libraries without any installations (PyTorch, SciKit learn, Numpy, Pandas, Matplotlib, etc.) and has free cloud service with GPU.

3.1 Data set

For any machine learning problem, the quality of the data plays a principal role as it drives the algorithm [88]. For this study we require a large data set that is spread uniformly across a range of inlet temperatures. Since experimental analysis provides a limited set of results, data for the training the algorithm will be developed from computational fluid dynamic analysis. By using CFD analysis, we can generate a regularly distributed data set that can used for training the machine learning algorithm. So computational fluid dynamic analysis is performed and it is highly validated against experimental data [2]. The data generated was part of Masters Thesis work by Ph.D. candidate Yang Chao [1]. A quick over view of the process is given here.

3.1.1 Computational Fluid Dynamic Analysis

A horizontal circular tube of diameter 6mm is modeled to study the flow characteristics of sCO_2 and this is based on experimental analysis performed by Dang et al. [2]. A 3D model is employed to account for the buoyancy, temperature and velocity distribution. To obtain a fully developed flow in the pipe, the entrance effects are eliminated by adding an adiabatic section of length 200 mm. Thermal effects are applied on rest of the pipe which is of length 500 mm.

Computational fluid dynamic analysis is performed in ANSYS fluent with k-ω SST as the turbulence model. The analysis process is repeated and wall temperature is obtained for a range of random heat flux and mass flux conditions. The data generated is organized into

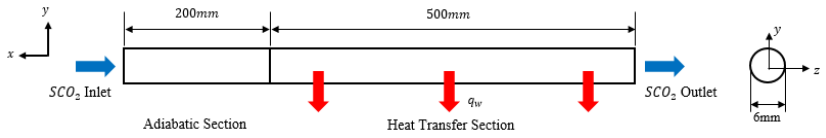

Figure 3.1 Geometry of the model [1]

10 files, this combination of ten files will be referred as the "Random Matrix". Table 3.1 shows the individual file names and their corresponding operating conditions. The pressure and diameter for all these runs is set to 8MPa and 6mm respectively.

Table 3.1 Random Matrix (RM) - operating conditions

File name	Bulk Temperature (T_b) (K)	Wall Temperature (T_w) (K)	Inlet Temperature (K)	Heat flux (q) (kW/m^2)	Mass Flux (G) (kg/m^2s)
RM_1	291 - 320	288 - 319	293 - 320	6	400
RM_2	303 - 333	295 - 329	303 - 333	24	400
RM_3	297 - 318	296 - 317	298 - 318	6	1000
RM_4	296 - 343	273 - 333	307 - 343	36	200
RM_5	306 - 333	301 - 329	307 - 333	24	600
RM_6	307 - 333	302 - 328	307 - 333	48	1000
RM_7	307 - 333	299 - 326	307 - 333	48	600
RM_8	305 - 333	300 - 330	306 - 333	24	800
RM_9	305 - 333	299 - 329	306 - 333	36	800
RM_{10}	304 - 333	301 - 331	305 - 333	12	600

3.2 Training, Validation and Test Set

The data for a well-posed problem is usually split into training and test set. The training set forms the largest part of the data and is used to fit (train) the model. For a fixed set of hyperparameters, the training data is used to fit the parameters of a model and the fitted model is used to predict on test data (new data). After the big data era, another set was introduced into the process called the validation set. This set is primarily used as part of the model development. We can change the hyperparamters and obtain different trained models and these model's performance can be checked using the validation data set. Using a validation set helps the model to provide an unbiased prediction on the test set. So for

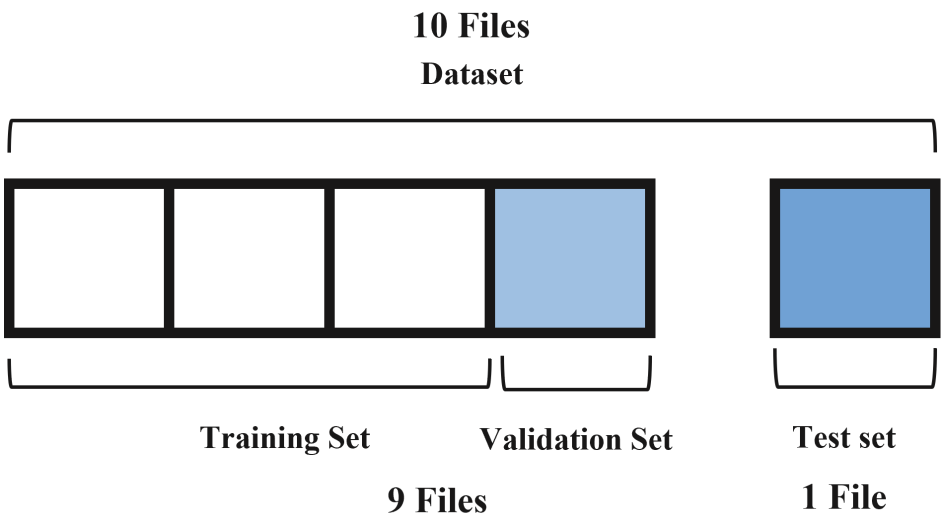

Figure 3.2 Splitting the Dataset

this study, the data set will be split into training, validation and test set. Although is not highly uncommon to do so, most of machine learning-sCO_2 papers have only used a training and validation set. The training, test and validation files are tabulated in table 3.2. We see that except for file RM_5 all the other files are considered as the training and validation set. The RM_5 corresponding to (heat flux) $q = 24kW/m^2$ and mass flux $G = 600kg/m^2s$ is taken as the test case. In the range of heat flux (q = [6, 12, 24, 36, 48]) values and mass flux values (G = [200, 400, 600, 800, 1000]), the $q = 24kW/m^2$ and $G = 600kg/m^2s$ are centered. The extreme cases (low heat flux - low mass flux and high heat flux - high mass flux) are weighed into the training and validation data so that model can learn the patterns in the data. Since the extreme cases are incorporated into the training data, the trained model should eventually be able to predict well when a complex input is given. Being able to predict these extreme cases will enable us to develop a model that can efficiently capture the non-linear heat transfer behaviour of supercritical carbon dioxide in the pseudo critical region.

Table 3.2 Training, Validation and Test data sets

		Feature	Feature	Feature	Target
	File name	Bulk Temperature (T_b) (K)	Heat flux (q) (kW/m^2)	Mass Flux (G) (kg/m^2s)	Wall Temperature (T_w) (K)
Training and Validation	RM_1	291 - 320	6	400	288 - 319
	RM_2	303 - 333	24	400	295 - 329
	RM_3	297 - 318	6	1000	296 - 317
	RM_4	296 - 343	36	200	273 - 333
	RM_6	307 - 333	48	1000	302 - 328
	RM_7	307 - 333	48	600	299 - 326
	RM_8	305 - 333	24	800	300 - 330
	RM_9	305 - 333	36	800	299 - 329
	RM_{10}	304 - 333	12	600	301 - 331
Test	RM_5	306 - 333	24	600	301 - 329

3.3 Input and Output parameters

The data generated from computational fluid dynamics contains four parameters that are of our interest and are to be sorted in terms of features and targets. As previously stated, the features will be the input parameters in the artificial neural network model and target is assigned to the output parameter. The input layer must contain independent variables that has significant influence on the variable that is to be predicted. From review of literature papers, the pressure, diameter, fluid bulk temperature, wall heat flux and inlet mass flux affects the heat transfer coefficient the most. For the generated dataset the pressure and diameter is fixed at 8MPa and 6mm respectively. Hence we consider bulk temperature (T_b), heat flux (q) and mass flux (G) to be our model's features. The heat transfer coefficient can be obtained from wall temperature, bulk temperature & wall heat flux and to reduce complexity, we take independent parameter wall temperature as the target value.

3.4 Scaling and Transforming the data

All the training and validation data needs to scaled before training. During the learning process, weights are randomly assigned to the input data. If certain input values are higher than others, this might affect how weights are assigned and this in turn would affect the

```
scaler_x = MinMaxScaler((-1,1))
X_t= scaler_x.fit_transform(X)

scaler_y = MinMaxScaler((-1,1))
y_t = scaler_y.fit_transform(y.reshape(-1,1))
```

Figure 3.3 Scaling function

learning process. So it is highly recommended to perform Feature Scaling so that all the input data has equal importance. Normalization and Standardization are used to scale the data. Normalization bounds the data between either [0,1] or [-1,1] and Standardization transforms the data such that it has zero mean and unity variance. MinMax Scaler, is a Scikit Learn [89] preprocessing function that scales data between [-1,1]. Figure.3.3 shows a code snippet where the X(features) and y(target) are scaled using MinMaxScaler and transformed using $fit.transform()$, another Scikit Learn [89] preprocessing function. The $fit.transform()$ is used only on the training data so that we can learn the scaling parameters of the training data. While working on the validation data we use only $transform()$. This is because we don't want to learn the new scaling parameters, instead we only want to transform the validation data using the scaling parameters from training data. Once the data is scaled, we also shuffle the rows of the data set using "numpy.random.shuffle" where Numpy is a library that performs mathematical functions.

3.5 Training the Model

To the train the data, the following must be established.

- Neural Network Definition
- Error Criterion
- Training and Validation Set
- Optimizer
- Epochs

A neural network module is defined using the torch.nn.module from Pytorch library [90]. Figure 3.4 shows the defined neural network. The parent class is constructed using the *int* function. The three $self.layers$ indicate the number of neurons and their connections in each layer of the artificial neural network. The first layer has 3 inputs and is connected to 10 neurons in the next layer. The $nn.Linear$ suggest that we have linear transformation applied and $nn.ReLU$ is the activation function applied. Similarly, the second $self.layer$ represents the hidden layers. There are initially 5 hidden layers defined with 10,30,50,50,50 number of neurons in each layer. Lastly, the output layer is defined with one output neuron and no activation function. By instantiating the model, the function definition is ran and its output is assigned to a variable.

```
import torch.nn as nn
from torchsummary import summary

class sCO2(nn.Module):
  def __init__(self):
    super(sCO2,self).__init__()
    self.layer1 = nn.Sequential(nn.Linear(3, 10),nn.ReLU(),)
    self.layer2 = nn.Sequential(nn.Linear(40, 120),nn.ReLU(),nn.Linear(120, 120),nn.ReLU(),
                                nn.Linear(120, 120),nn.ReLU(),nn.Linear(120, 120),nn.ReLU(),)
    self.layer3 = nn.Sequential(nn.Linear(120,1),)


  def forward(self,x):
    out = self.layer1(x)
    out = self.layer2(out)
    out = self.layer3(out)


    return out

## Instantiating the model
model = sCO2()
summary(model,(1,3))
```

Figure 3.4 Scaling function

The error criterion defined here of this study is Mean Squared Error (MSE). Scikit learn library has several pre-programmed performance metrics that can be used for different types of problems. For a regression problem, we have Mean Squared Error MSE, Root Mean Squared Error $RMSE$, Mean Absolute Percent Error $MAPE$, R^2, etc. Mean Squared Error given by eq.**??** is the squared residuals of the true value and predicted value. The training data and validation data that contains elements from 9 files, is split using "$Train.test.split$"

```
# zero the parameter gradients
optimizer.zero_grad()

# Forward pass
outputs = model(X_train)
loss = criterion(outputs, y_train)
# Backward and optimize
loss.backward()
optimizer.step()

# Save training loss
train_losses[it] = loss.item()

# Saving validation loss
output = model(X_val)
loss_val = criterion(output,y_val)
val_losses[it] = loss_val.item()
```

Figure 3.5 Learning Process

function. This function splits the data into two sets. Although the name suggests that it is splitting the data into training and test set, note that we use it to split the data into training and validation. The 9 files from the Random Matrix has a total of 23,550 data points. From this 10% of data points are assigned to the validation set(2,355 data points) using the "train.test.split" function. It is essential to allocate a fixed number for *random.state* so that we can reproduce the randomness. Every time we run the code, we want the algorithm to pick the same random numbers. Once the data is split, the learning process of the neural network is coded. Figure 3.5 shows the block of code used to define the feed forward and back propagation process. As mentioned previously, the learning process involves an optimizer and a cost function. Firstly the gradients of the inputs are zeroed by using *optimizer.zero.grad*(). The inputs are now passed into the neural network model and the loss criterion is estimated. This leads to the next step back propagation, the *loss.backward*() is used perform this task.

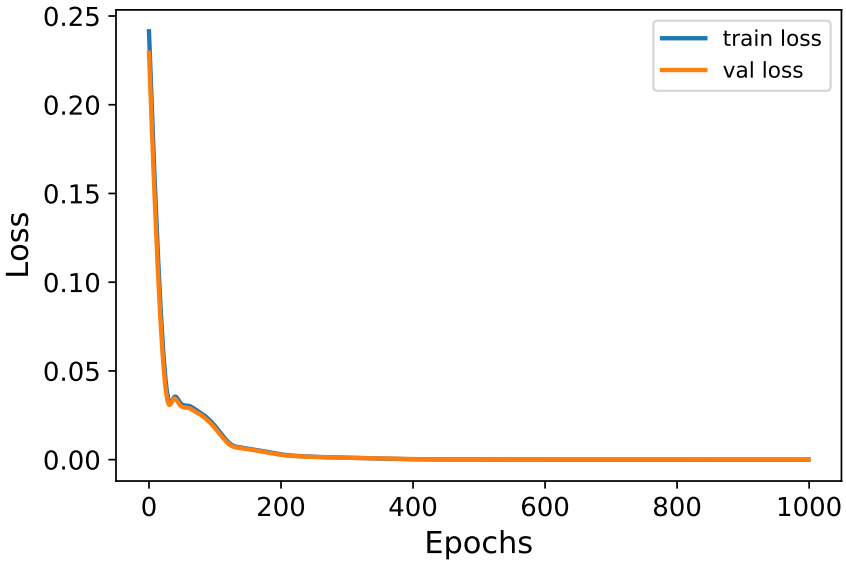

Figure 3.6 Training and Validation Loss

Lastly, we use *optimizer.step*() which a function that updates the parameters of the process. The Adam optimizer from PyTorch library is used. This whole process is repeated for a number of epochs. An epoch is completed when the learning algorithm passes through the entire training and validation data set once. For this study, we use 1000 epochs. We also record the training and validation errors to keep an eye on over-fitting or under-fitting. The training and validation loss plotted for each epoch is shown in fig.3.6. During training, the losses should decrease as the epochs are increased. With the initial parameters defined we obtained a trained and validated model C1 (base model). A summary of the model is given in table 3.3. The parameters in the table were taken as the starting point based on trail & error, literature and the fact that they are most commonly used parameters for artificial neural network models. Using this model, a neural network is developed and will be trained and validated on the training data set and validation data set. Based on error (MSE) produced by the base model on the test case, the model's efficiency will be improved by hyperparameter tuning.

Table 3.3 Summary of base model C1

Model	Base model C1
Input parameters	3 - [T_b, q, G]
Hidden layer	5
Number of neurons in each layer	10-30-30-30-50
Output parameter	1 - [T_w]
Activation Function	ReLU
Optimizer	Adam
Loss	MSE
Epochs	1000

4 Results

With the trained and validated neural network model, an initial prediction is made on the test data set ($q = 24KW/m^2$ and $G = 200kg/m^2s$). Note that the test data is kept hidden from the model during training and validation. The prediction is made on the unseen data using the scaling parameters obtained from the trained model. Figure 4.1 shows the wall temperature predicted using the neural network model (C1) compared against the computational fluid dynamics data. In the x-axis, we have bulk temperature and in y -axis, we have wall temperature. The Mean Squared error obtained from this model is 1.41 K^2. In the figure, we can see that around bulk temperature of 308 K, there is a small peak in wall temperature and C1 model is unable to capture it. Also, after $T_b > 320$K, the neural network model starts to overestimate the wall temperature. This over-prediction can be associated to the fact that there are gaps in the computational fluid dynamics data. For a good model, the mean squared error should be close to zero. Although the training and validation losses were low for this model, the prediction on the test data is producing a high mean squared error. Therefore, we need to tune the model further by changing the hyperparameters such that the loss on test cases become relatively lower. This process is called Hyperparameter tuning.

4.1 Hyperparameter Study

In this section, the hyperparameters of a neural network model are varied to see which parameters will produce a better model. The error on test data is compared for each of these tuned models. For a neural network model, the main hyperparameters are - scaling functions, number of hidden layers, number of neuron in layers, dropout, optimizer, learning rates and weight decay. There is no procedure or order defined to vary the hyperparameters. So, in this study each of the listed hyperparameters will be varied in the base model (C1) and in the end and a procedure is established for hyperparameter tuning.

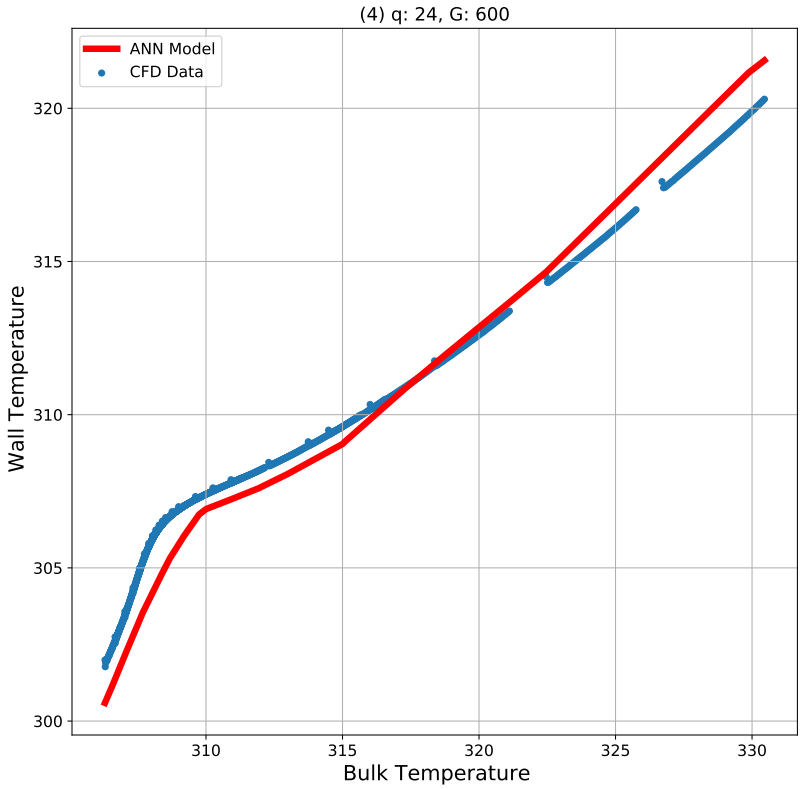

Figure 4.1 Wall temperature predicted by ANN base model (C1) compared against the CFD data (true values)

4.1.1 Scaling Functions

The MinMaxScaler that was initially considered for the scaling the input data in base model is the most widely recognised function used to standardize information [91]. In order to cross check, other functions such as StandardScaler, RobustScaler and QuantileTranformer are used instead in the base model. Training, Validation error, mean squared error and the maximum error (higest deviation from the true values) on the test case are tabulated in table 4.1. Although RobustScaler function has the least error on the test set, MinMaxScaler has the least training and validation errors. The presence of outliers in dataset negatively influences the mean and variance in such cases the RobustScaler preforms better. This is because the function removes the median and scales the data to the interquartile range(between 25th

Table 4.1 Hyperparameter tuning - Scaling Functions

Function	Training error K^2	Validation error K^2	Test error K^2	Max error in test case K
MinMaxScaler	2.87E-05	3.24E-05	1.41	1.86
StandardScaler	3.65E-04	3.12E-04	1.10	2.76
RobustScaler	5.37E-04	4.46E-04	0.15	0.83
QuantileTransformer	1.48E-04	1.36E-04	0.93	2.31

quantile and 75th quantile)[92]. Figures 4.2 - 4.4 depicts the neural network models with different scaling function making prediction on the test data set. The QuantileTransformer that scales the data under a normal distribution, does not consider the outliers. Since our data has outliers in the test set, it fails to predict well. Note that the maximum error on the test set was obtained when the training data was scaled using StandardScaler. The drawback of StandardScaler is if a feature has high variance, it might lead the estimator to incorrectly learn the features that have lower variance [93]. Comparing the prediction of RobustScaler case on test data (fig.4.3) with the MinMaxScaler case (fig.4.1), we visually see that the former model is capturing the initial peak well($T_b < 308K$). But looking at the training and validation errors of the two models, the MinMaxScaler has the lowest loss and is considered to be the better scaling option for this dataset.

4.1.2 Number of Layers

Choosing the number of layers in the neural network plays a substantial role in determining the complexity of the model. The increase in number of hidden layers might increase the accuracy but also might lead to over-fitting and a predominant increase in computational cost. Initially for the base model we start with 5 layers and since this is already a deeper neural network, we should pay attention to the training and validation error while increasing the number of layers. Over-fitting will occur when we fit a complex model to a simple problem. With the base model, two layers are added each time and prediction on the test set is obtained. Figures 4.5 and 4.6 shows the C2 and C3 model prediction respectively. The model C2 with 7 hidden layers under predicts the CFD data But when we increase the

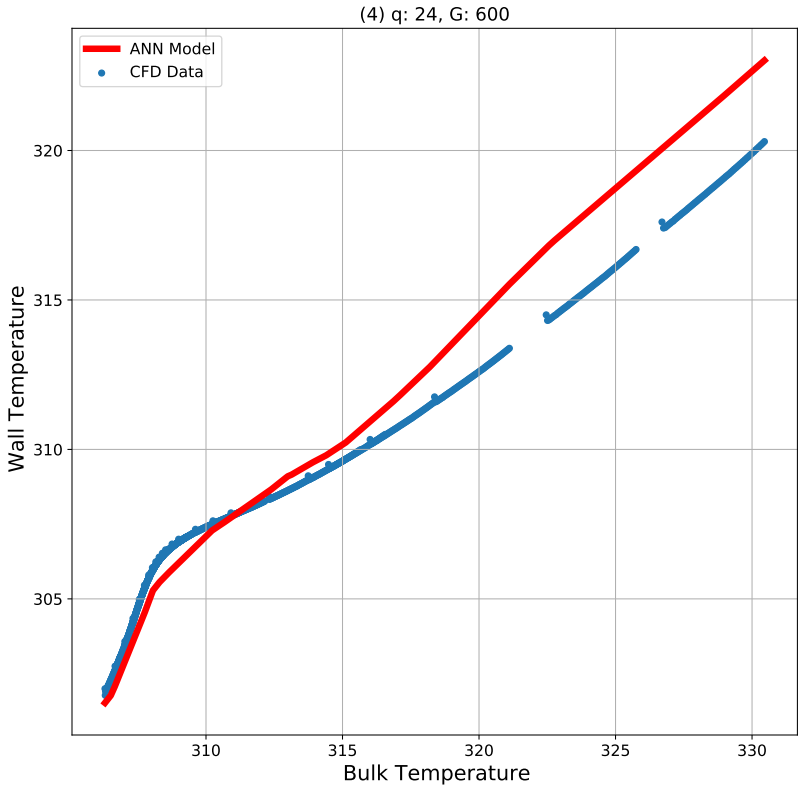

Figure 4.2 Wall temperature predicted by ANN model StandardScaler function compared against the CFD data

number of layers to 9 layers, the model performs relatively better.

The training and validation loss was plotted while running these two cases. As the model is trained, both training and validation errors should decrease, but at some point the validation error would increase and this would indicate that the model is starting to over-fit. Figure 4.7 depicts the training and validation loss plotted while training C3 model. The model has 9 hidden layers and in the figure we can see the errors are tending to zero. This essentially means that the model is not over-fitting. It is not clear why the C2 model with 7 layers has a larger deviation from true value but the efficient number of hidden layers can only be estimated through trial and error method. Although it took only few minutes for these deeper models to train and predict, the C3 model took more time to train compared

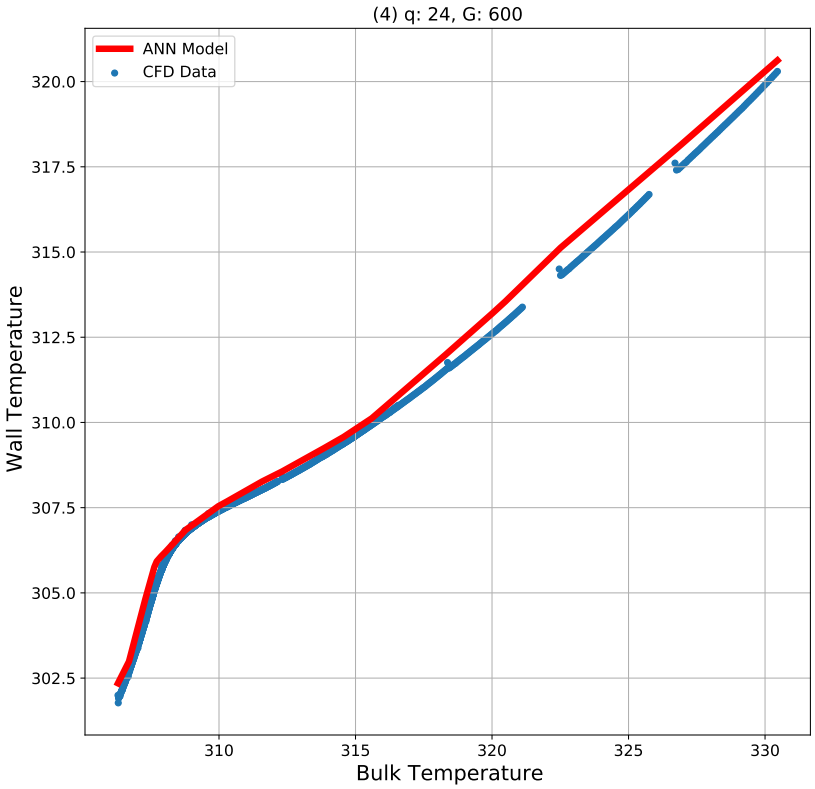

Figure 4.3 Wall temperature predicted by ANN model RobustScaler function compared against the CFD data

to other models.

4.1.3 Number of Neurons

The number of neurons in the hidden layer have a similar effect as the number of hidden layers. Increasing the number of neurons comes with the penalty of over-fitting. The base model has 5 layers with 10, 30, 30, 30 and 50 number of neurons in each layer. This was decided using trail and error method. We can increase the number of neuron in the 5 layers and see how the model behaves to tuning this hyperparameter. For model C4, the number of neurons in the hidden layers are doubled (20, 60, 60, 60 and 100) and for model C5, the number of neurons are tripled (30, 90, 90, 90, 90 and 150).

Table 4.3 shows the training and validation losses obtained for model C4 and C5. While

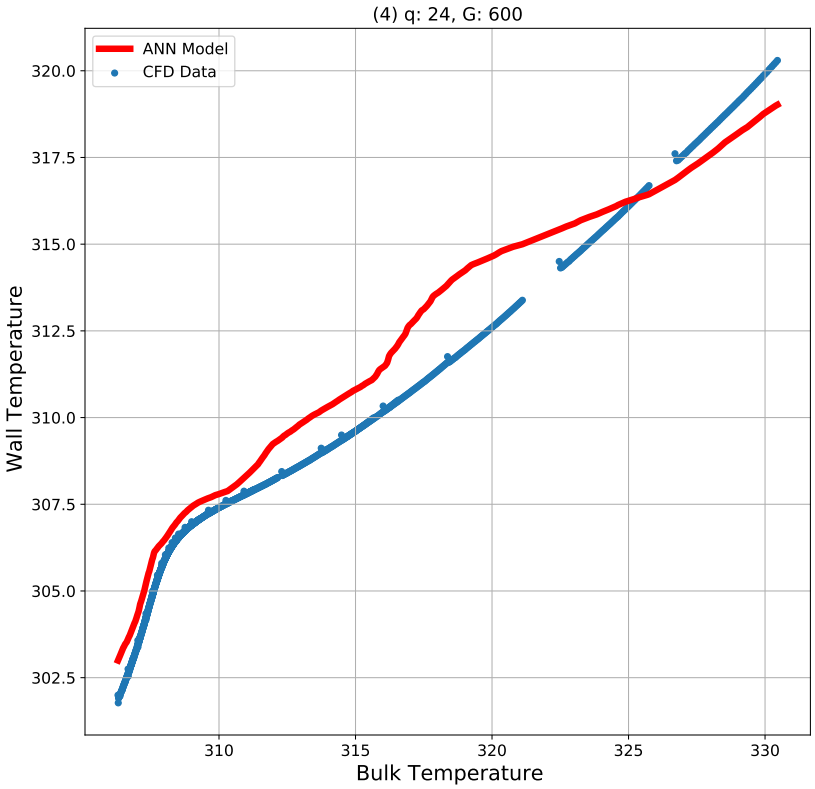

Figure 4.4 Wall temperature predicted by ANN model QuantileTransformer function compared against the CFD data

the training error and validation error of model C5 is lesser than model C4, the errors on the test set are higher. The C5 model has learned the parameters too well leading to over-fitting. Figures 4.8 and 4.9 shows the wall temperature prediction of the models compared against the CFD data. We see that model C5 under-predicts at $T_b < 315K$ and $T_b > 325K$.

4.1.4 Dropout

Regularization of the model is performed to avoid over-fitting. L1, L2 penalty and Dropout are some of the general regularization methods. L1 and L2 methods adds a penalty to the weight terms so that some features will have lesser influence on the final output. Dropout is the most popular technique against over-fitting. In this method, some neurons and connections are dropped randomly during training. The model is trained by dropping

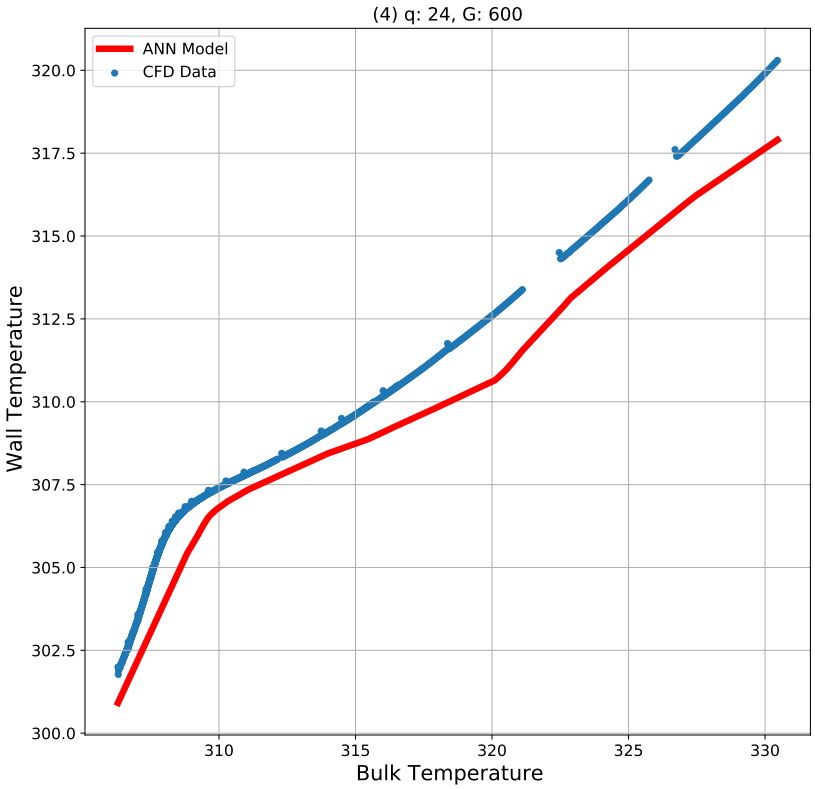

Figure 4.5 Wall temperature predicted by ANN model (C2) compared against the CFD data

different batches of neurons, this helps in reducing the complexity of the neural network and the process is repeated until ideal parameters are obtained [94]. "nn.Dropout" is used in the model definition, it receives a float data type (p) that represents the percent of neurons to dropout. To evaluate Dropout hyperparameter, we use model C5 which had 30,90,90,90,90 and 150 neurons in the hidden layers. The higher number of neurons increased complexity of this neural network. A dropout value of p = 0.2 is implemented in the last hidden layer and this enables the algorithm to drop 20% of the neurons in the last hidden layer. Similarly for another case, dropout value of p = 0.5 is implemented in the last layer. Figures 4.10 and 4.11 shows the neural network models C5_d1 and C5_d2 prediction on test data. Although the training error and validation error are different for the cases, we see that the MSE and max

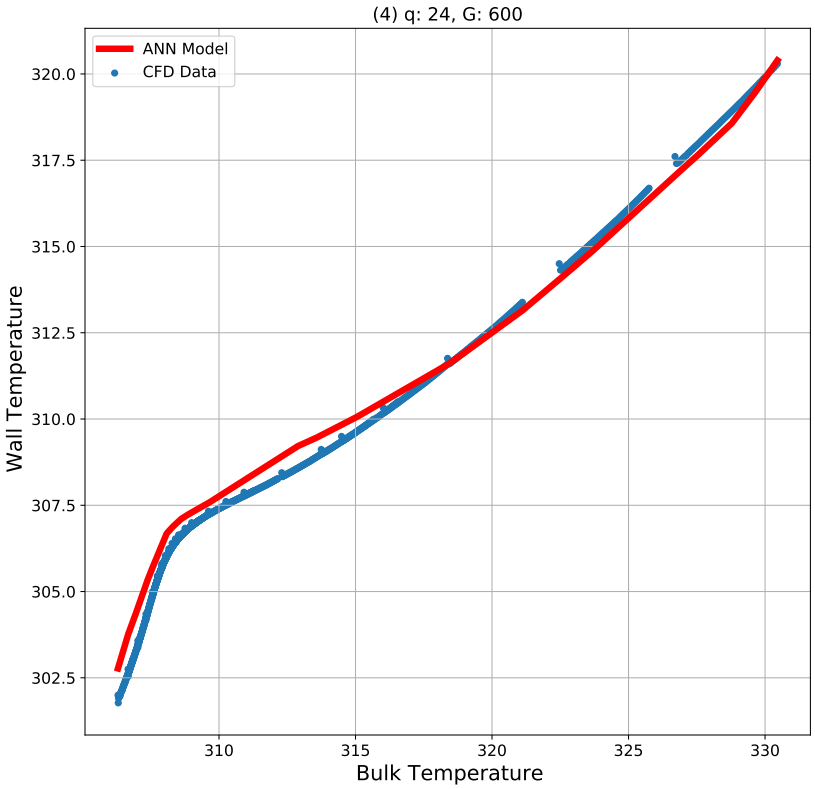

Figure 4.6 Wall temperature predicted by ANN model (C3) compared against the CFD data

error on the test data are approximately equal. Although, the different dropout percentage is not affecting the model's prediction on the test data set, using dropout in the last layer is enabling the model to correctly predict the wall temperature after $T_b > 315K$.

4.1.5 Optimizer

The learning process of the neural network is dependent on the optimizer. Gradient descent is one of the most popular algorithms used for optimization. It iteratively calculates the the local minimum of the cost function by using the negative gradient of the initial position. It has many variants that are highly efficient. Ruder et al. [95] evaluates Batch Gradient Descent, Stochastic Gradient Descent (SGD) and Mini Batch Gradient Descent, Adagrad, RMSprop, etc. which are all different variants of Gradient Descent. "Pytorch.optim" [96]

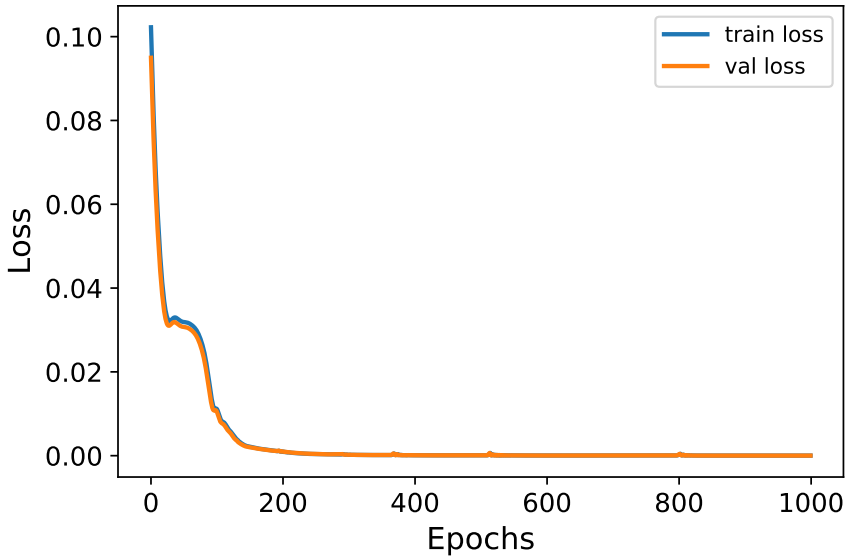

Figure 4.7 Training and Validation loss for Case 3 (9 Hidden Layers)

Table 4.2 Hyperparameter tuning - Number of Layers

Case	Training error K^2	Validation error K^2	Test error K^2	Max error in test case K
Base Case (C1) (5 hidden layers)	2.87E-05	3.24E-05	1.41	1.86
Case 2 (C2) (7 hidden layers)	1.02E-04	1.11E-04	2.03	2.39
Case 3 (C3) (9 hidden layers)	2.35E-04	2.78E-04	0.36	1.22

package has various optimization algorithms. Adam (adaptive moment estimation) optimizer which combines the advantages of two optimization algorithms (Adagrad and RMSprop) is being widely used to solve machine learning problems. Derya [97] describes the working of different optimization algorithms and their pros and cons. The Adam algorithm is appropriate for problems with noisy data and sparse gradients. Because Adam is considered to be the better algorithm for the current dataset, we do not change or tune this hyperparameter. But for someone with little domain knowledge predicting on a new problem, it is highly encouraged to implement different optimization algorithms and choose the best algorithm by trail and error.

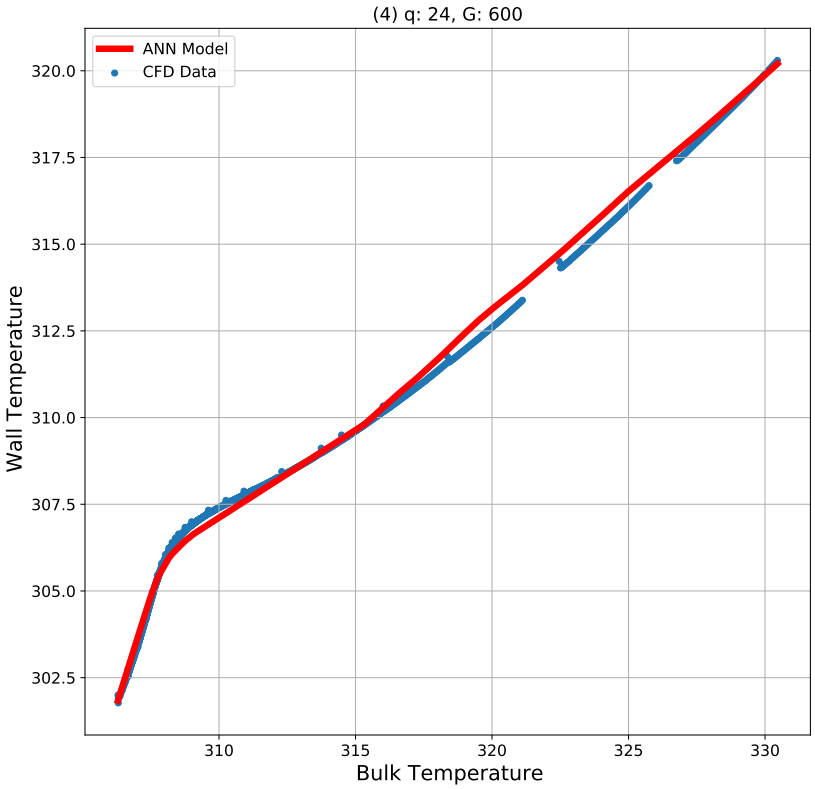

Figure 4.8 Wall temperature predicted by ANN model (C4) compared against the CFD data

4.1.6 Learning Rate and Weight Decay

Although, the optimizer is kept constant we can change the parameters of an optimizer. The main parameters in Adam optimizer are learning rate, weight decay, betas and epsilon [98]. The learning rate parameter by default is set to 0.001. It affects how quickly a model can converge towards the local minima. A very high learning rate will cause the model to converge quickly at a pseudo minima and a lower learning rate will lead the process to fluctuate. In model C5, we increase the model learning rate to 0.01 and see how the model performs. Figure 4.13 show the model's (L1) prediction on test data set. We see that the model is able to capture the peak well but under predicts near the gaps in CFD data.

Typically weight decay can be implemented in Adam optimizer directly and by default

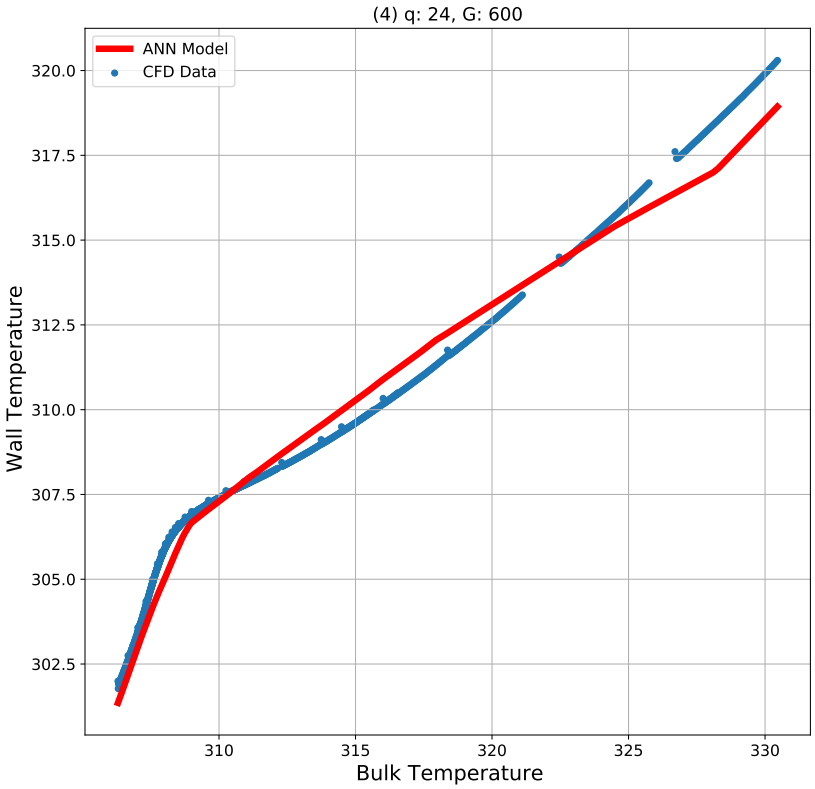

Figure 4.9 Wall temperature predicted by ANN model (C5) compared against the CFD data

the value is set to zero. But recently, there is an update on the weight implementation in Adam optimizer. Loshchilov et al. [99] introduced the AdamW optimizer that uses a modified weight decay formula in the gradient update. Weight decay is a introduced to reduce the complexity of the model and prevent over-fitting. So we use AdamW optimizer with a weight decay parameter equal = 0.1 in the C5 model and see if it is able to reduce the mean squared error.

The table 4.5 summarizes the errors for Models L1(learning rate model) and Wd(weight decay model). Increasing the learning rate has reduced the mean squared error on the test set. A higher learning rate is helping the model to capture the first peak in the test data. There isn't significant difference between the model C5 and Wd. Increasing penalty on the

Table 4.3 Hyperparameter tuning - Number of Neurons

Case	Training error K^2	Validation error K^2	Test error K^2	Max error in test case K
Base Case (C1) (5 hidden layers)	2.87E-05	3.24E-05	1.41	1.86
Case 4 (C4) (x2 neurons)	7.34E-06	6.61E-06	0.12	0.53
Case 5 (C5) (x3 neurons)	5.82E-06	5.00E-06	0.38	1.42

Table 4.4 Hyperparameter tuning - Dropout (*p = input for dropout)

Case	Training error K^2	Validation error K^2	Test error K^2	Max error in test case K
Case 5 (C5) (x3 neurons)	5.82E-06	5.00E-06	0.38	1.42
Case 5d1 (C5_d1) (p=0.2)	4.03E-04	3.94E-04	0.56	1.24
Case 5d2 (C5_d2) (p=0.5)	9.49E-04	9.73E-04	0.76	1.42

weight decay from zero to 0.1 did not produce a better model. Since only weight decay is being added to the model, there isn't significant improvement.

The tuning of the main hyperparameters in a neural network model was discussed. The activation function and number of epochs can also be varied to obtain an optimal fit. It was observed that changing certain parameters such as the increasing the number of layers and adding dropout to model C5 enabled the model to capture the regions where we had gaps in the computational fluid dynamics data. These variations helped the model predict well on test data after the peak. On the other hand, changing other hyperparameters such as the scaling function and learning rates allowed the model to predict the peak in test data accurately. Also note that just by changing one hyperparameter - number of neurons in the hidden layers, the trained model C4 was predicting comparatively better with an error of 0.05. This leads us to the next part where more than one hyperparameter is changed to obtain an efficient model.

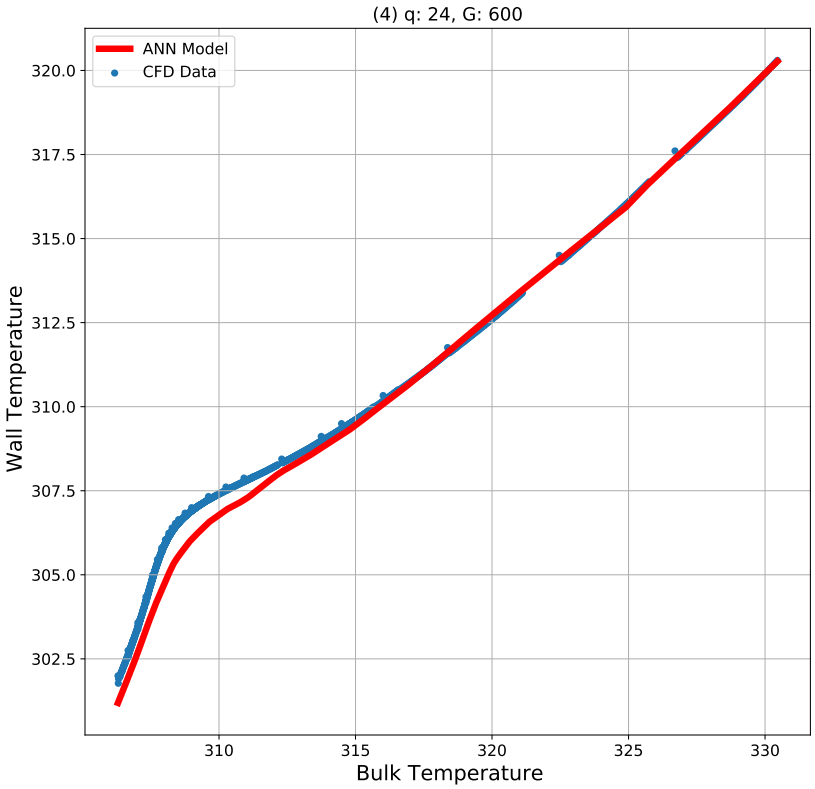

Figure 4.10 Wall temperature predicted by ANN model (C5_d1) compared against the CFD data

4.2 Final Model

A combination of hyperparameters are tuned to obtain a final model. In the base model C1, two layers are added and AdamW implemented with learning rate of 0.01 and weight decay of 0.1. The final model predicts on the test data with a mean squared error of 0.08 K^2 and the maximum error is 0.65 K. The scaling parameters and model definition from this model are saved as "pickle files". A pickle file makes it easier to store and upload the model parameters for future use. Figure 4.14 depicts the wall temperature predicted for the the test case compared against CFD data. The gaps in the CFD data and the peak in bulk temperature is captured well by this model. Reviewing the mean squared error produced by the base model and final model on the test set, we can observe that there is 94% decrease

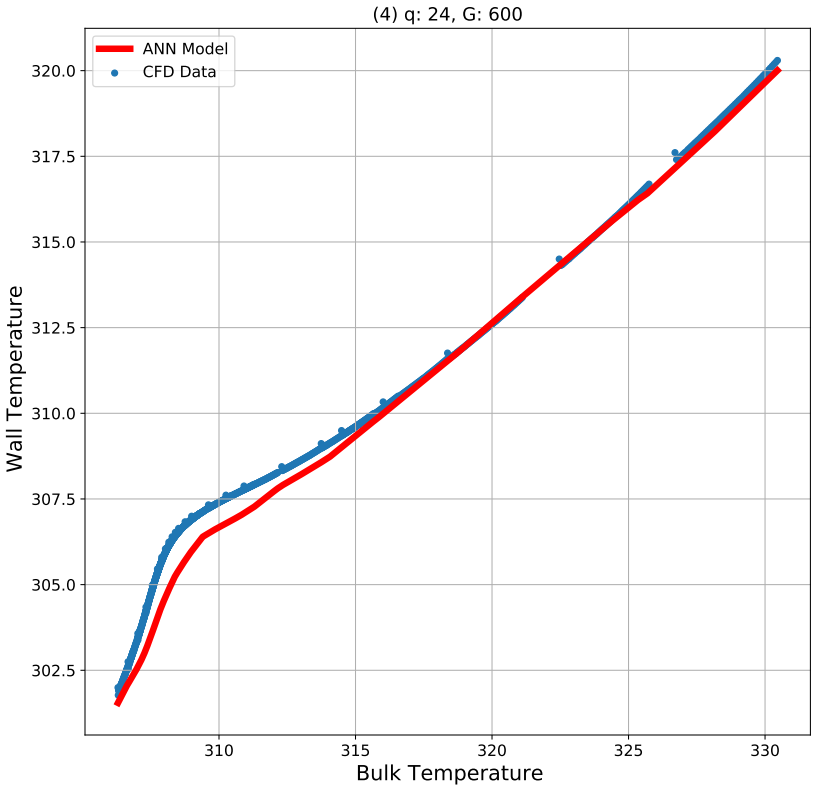

Figure 4.11 Wall temperature predicted by ANN model (C5_d2) compared against the CFD data

in error. Using hyperparameter tuning we were able to obtain a better model.

The final model definition, scaling parameters and the trained model can be found in the google drive folder (ANN Files). These files can be used by anyone to predict the wall temperature of supercritical carbon dioxide held at 8MPa in a horizontal tube of diameter 6mm under cooling conditions using heat flux, mass flux and bulk temperature conditions for a inlet temperature ranging between 25°C-30°C.

Table 4.5 Hyperparameter tuning - Learning Rate and Weight Decay

Case	Test error K^2	Max error in test case K
Case 5 (C5) (x3 neurons)	0.38	1.42
Case L (Learning rate) (lr=0.01)	0.12	1.07
Case Wd (weight decay) (wd = 0.1)	0.36	1.61

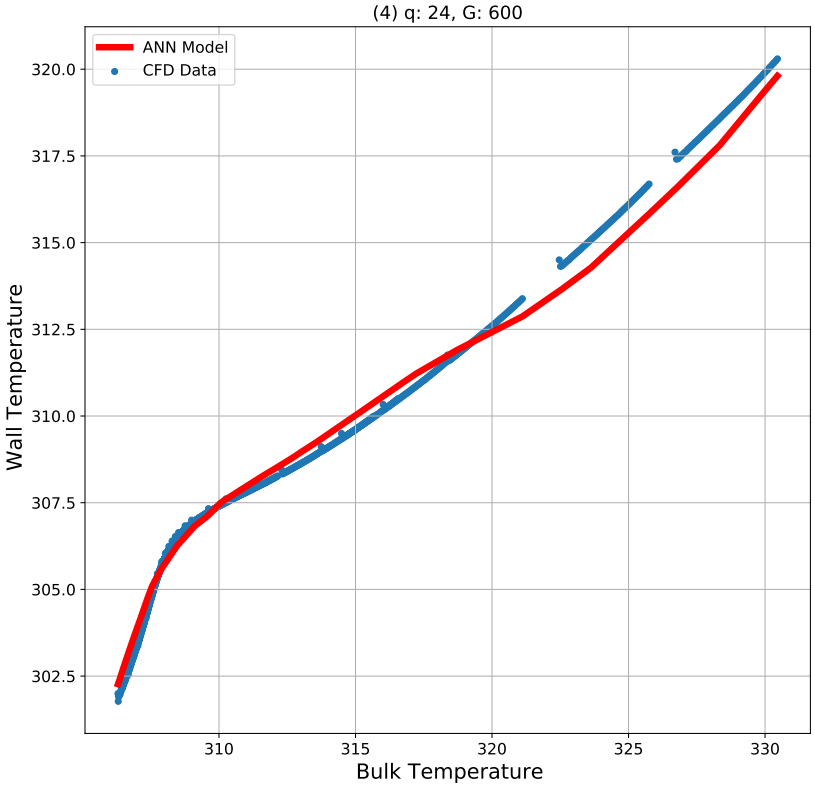

Figure 4.12 Wall temperature predicted by ANN model (L1) compared against the CFD data

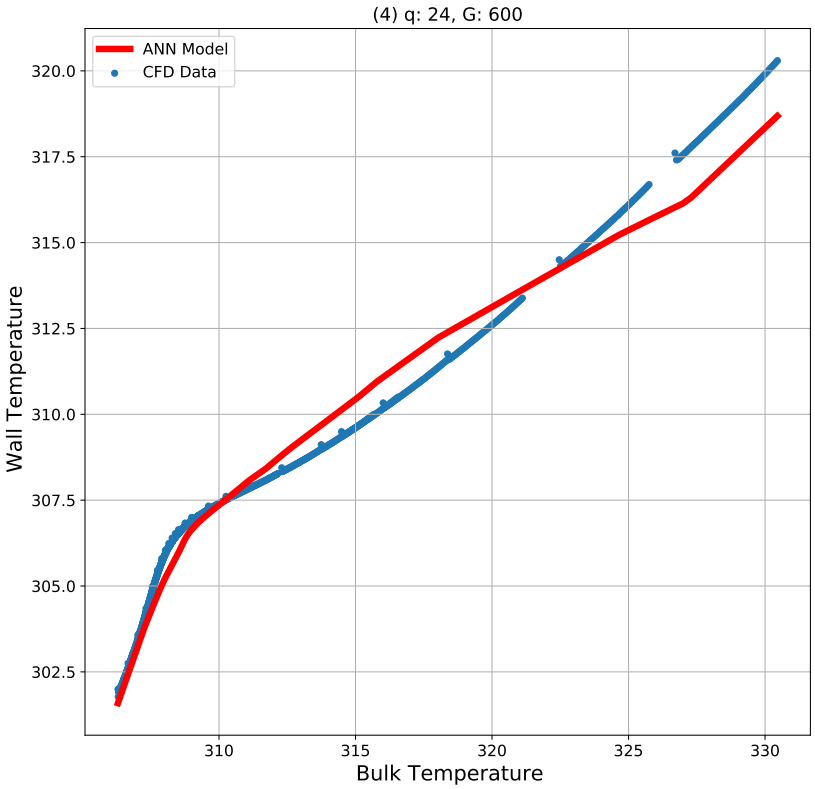

Figure 4.13 Wall temperature predicted by ANN model (Wd) compared against the CFD data

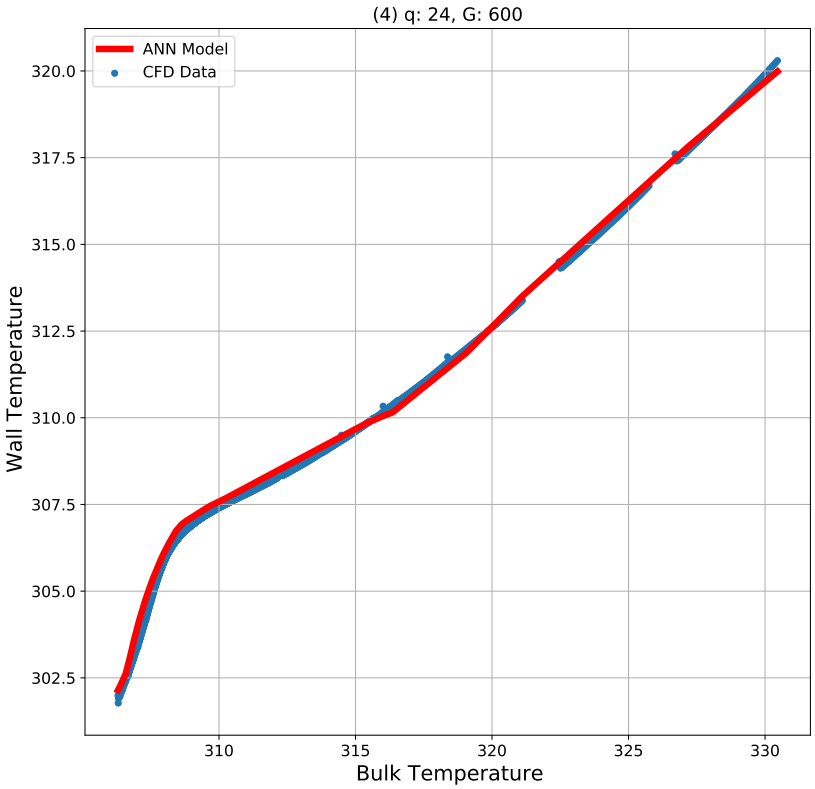

Figure 4.14 Wall temperature predicted by ANN (Final Model) compared against the CFD data

5 Conclusion and Future Work

This study aimed to develop an artificial neural network model trained on highly validated computational fluid dynamic analysis data. The operating pressure for CFD data is 8MPa which is near the critical pressure (7.39 MPa). The heat flux, mass flux and bulk temperature ranges from $6kW/m^2 - 48kW/m^2$, $200kg/m^2s - 1000kg/m^2s$ and $291K - 333K$ respectively. The computational fluid dynamic data has approximately 25,000 data points out of which 21,191 data points was used training and 2,355 data points was used to validate the model. The trained model was tested on high heat flux and mass flux data corresponding to q = $24kW/m^2$ and G = $600kg/m^2s$. Initially the model C1 produced a high mean squared error of 1.41 K^2. Using hyperparameter tuning, a successful attempt was made to reduce the test error while also not over-fitting on the training data. The significant hyperparameters in an artificial neural network model were varied and it was found that some of hyperparameters such as number of hidden layers and dropout helped the model predict near the gaps well and other hyperparameters such as earning rate and weight decay helped the model predict the peak in the data effectively. With this knowledge, a final artificial neural network model was developed by varying a combination of hyperparameters. The final model has 7 hidden layer with 10-30-30-30-30-30-50 neurons in each layer, AdamW as the optimizer with a learning rate of 0.001 and a weight decay of 0.1. The final model produced a mean squared error of 0.08 K^2 on the test case. It only took a few minutes to train, validate and test the final model and so neural network are computationally cost efficient compared to traditional approaches. The final model's scaling parameters, model definition and trained model can be found in the google drive that is accessible to all. A code is also provided in Appendix for anyone who wants to use the final model to make predictions on new data.

The study shows that artificial neural networks are highly capable of capturing the non-linearity in heat transfer behaviour of supercritical carbon-dioxide. With a strong understanding on the working and hyperparameter tuning of artificial neural networks, more efficient models can be developed. Future works would include training the final model on

different data sets corresponding to different pressure ranges. A physics informed neural network can used explored as it would employ differential equations and mathematical models into machine learning to create powerful neural network models. Further, since there is very less research on the uncertainty analysis of the artificial neural network predictions, Bayesian neural network which is a combination of artificial neural network and Bayesian inference must be explored.

REFERENCES

[1] Chao, Y., "Analysis of Local Convection Heat Transfer Rates of Supercritical Carbon Dioxide in Tubes," 2020.

[2] Dang, C., and Hihara, E., "In-tube cooling heat transfer of supercritical carbon dioxide. Part 1. Experimental measurement," *International Journal of Refrigeration*, Vol. 27, No. 7, 2004, pp. 736–747. https://doi.org/10.1016/j.ijrefrig.2004.04.018.

[3] Fang, X., and Xu, Y., "Modified heat transfer equation for in-tube supercritical CO2 cooling," *Applied thermal engineering*, Vol. 31, No. 14-15, 2011, pp. 3036–3042.

[4] Cooper, A. I., "Polymer synthesis and processing using supercritical carbon dioxide," *Journal of Materials Chemistry*, Vol. 10, No. 2, 2000, pp. 207–234.

[5] Murga, R., Ruiz, R., Beltran, S., and Cabezas, J. L., "Extraction of natural complex phenols and tannins from grape seeds by using supercritical mixtures of carbon dioxide and alcohol," *Journal of Agricultural and Food Chemistry*, Vol. 48, No. 8, 2000, pp. 3408–3412.

[6] Dadashev, M., and Stepanov, G., "Supercritical extraction in petroleum refining and petrochemistry," *Chemistry and technology of fuels and oils*, Vol. 36, No. 1, 2000, pp. 8–13.

[7] Raventós, M., Duarte, S., and Alarcón, R., "Application and possibilities of supercritical CO2 extraction in food processing industry: an overview," *Food Science and Technology International*, Vol. 8, No. 5, 2002, pp. 269–284.

[8] Park, H. S., Lee, H. J., Shin, M. H., Lee, K.-W., Lee, H., Kim, Y.-S., Kim, K. O., and Kim, K. H., "Effects of cosolvents on the decaffeination of green tea by supercritical carbon dioxide," *Food Chemistry*, Vol. 105, No. 3, 2007, pp. 1011–1017.

[9] Dostal, V., Driscoll, M. J., and Hejzlar, P., "A supercritical carbon dioxide cycle for next generation nuclear reactors," 2004.

[10] Conboy, T., Wright, S., Pasch, J., Fleming, D., Rochau, G., and Fuller, R., "Performance characteristics of an operating supercritical CO2 Brayton cycle," *Journal of Engineering for Gas Turbines and Power*, Vol. 134, No. 11, 2012.

[11] "Industrial applications of supercritical fluids: A review," *Energy*, Vol. 77, 2014, pp. 235–243. https://doi.org/https://doi.org/10.1016/j.energy.2014.07.044.

[12] Bringer, R., and Smith, J., "Heat Transfer in the Critical Region," *AlChE*, Vol. 3, No. 1, 1957, pp. 49–55.

[13] Krasnoshchekov, E., Kuraeva, I., and Protopopov, V., "Local Heat transfer of Carbon Dioxide at Supercritical Pressure Under Cooling Conditions," *High Temperature*, Vol. 7, No. 5, 1970, pp. 922–930.

[14] Petukhov, B. S., and Kirillov, V. V., "About Heat Transfer at Turbulent Fluid Flow in Tubes (in Russian)," *Thermal Engineering*, , No. 4, 1958, pp. 63–68.

[15] Krasnoshchekov, E., Protopopov, V., Parkhovnik, I., and Silin, V., "Some Results of an Experimental Investigation of Heat Transfer to Carbon Dioxide at Supercritical Pressure and Temperature Heads of up to 850 C," *High Temperature*, Vol. 9, No. 5, 1972, pp. 992–995.

[16] Gnielinski, V., "New equations for heat and mass transfer in the turbulent flow in pipes and channels," *NASA STI/recon technical report A*, Vol. 41, No. 1, 1975, pp. 8–16.

[17] Petukhov, B., Kurganov, V., and Gladuntsov, A., "Heat transfer in turbulent pipe flow of gases with variable properties," *Heat Transfer-Soviet Research*, Vol. 5, No. 4, 1973, pp. 109–116.

[18] Pitla, S. S., Groll, E. A., and Ramadhyani, S., "New correlation to predict the heat transfer coefficient during in-tube cooling of turbulent supercritical CO2," *International Journal of Refrigeration*, Vol. 25, No. 7, 2002, pp. 887–895. https://doi.org/10.1016/s0140-7007(01)00098-6.

[19] Liao, S. M., and Zhao, T. S., "Measurements of Heat Transfer Coefficients From Supercritical Carbon Dioxide Flowing in Horizontal Mini/Micro Channels," *J of Heat Transfer-Transactions of the ASME*, Vol. 124, No. 3, 2002, pp. 413–420. https://doi.org/10.1115/1.1423906.

[20] Jackson, J. D., and Hall, W. B., *Turbulent Forced Convection in Channels and Bundles: Theory and Applications to Heat Exchangers and Nuclear Reactors*, Hemisphere Publishing Corporation, New York, 1979, chapter and pages, pp. 613–640.

[21] Lopes, N. C., Chao, Y., Dasarla, V., Sullivan, N. P., Ricklick, M. A., and Boetcher, S. K., "Comprehensive Review of Heat Transfer Correlations of Supercritical CO2 in Straight Tubes Near the Critical Point: A Historical Perspective," *Journal of Heat Transfer*, Vol. 144, No. 12, 2022, p. 120801.

[22] Petrov, N., and Popov, V., "Heat-transfer and resistance of carbon-dioxide being cooled in the supercritical region," *Thermal Engineering*, Vol. 32, No. 3, 1985, pp. 131–134.

[23] Yoon, S. H., Kim, J. H., Hwang, Y. W., Kim, M. S., Min, K., and Kim, Y., "Heat transfer and pressure drop characteristics during the in-tube cooling process of carbon dioxide in the supercritical region," *International journal of refrigeration*, Vol. 26, No. 8, 2003, pp. 857–864.

[24] Baskov, V., Kuraeva, I., and Protopopov, V., "Heat-transfer with turbulent-flow of a liquid at supercritical pressure in tubes under cooling conditions," *High Temperature*, Vol. 15, No. 1, 1977, pp. 81–86.

[25] Petrov, N., and Popov, V., "Heat transfer and hydraulic resistance with turbulent flow in a tube of water at supercritical parameters of state," *Thermal Engineering*, Vol. 35, No. 10, 1988, pp. 577–580.

[26] Huai, X., Koyama, S., and Zhao, T., "An experimental study of flow and heat transfer of supercritical carbon dioxide in multi-port mini channels under cooling conditions," *Chemical engineering science*, Vol. 60, No. 12, 2005, pp. 3337–3345.

[27] Huai, X., and Koyama, S., "Heat transfer characteristics of supercritical CO2 flow in small-channeled structures," *Experimental Heat Transfer*, Vol. 20, No. 1, 2007, pp. 19–33.

[28] Son, C.-H., and Park, S.-J., "An experimental study on heat transfer and pressure drop characteristics of carbon dioxide during gas cooling process in a horizontal tube," *International Journal of Refrigeration*, Vol. 29, No. 4, 2006, pp. 539–546.

[29] Kuang, G., Ohadi, M., and Dessiatoun, S., "Semi-empirical correlation of gas cooling heat transfer of supercritical carbon dioxide in microchannels," *HVAC&R Research*, Vol. 14, No. 6, 2008, pp. 861–870.

[30] Oh, H.-K., and Son, C.-H., "New correlation to predict the heat transfer coefficient in-tube cooling of supercritical CO2 in horizontal macro-tubes," *Experimental Thermal and Fluid Science*, Vol. 34, No. 8, 2010, pp. 1230–1241.

[31] Ehsan, M. M., Guan, Z., and Klimenko, A., "A comprehensive review on heat transfer and pressure drop characteristics and correlations with supercritical CO2 under heating and cooling applications," *Renewable and Sustainable Energy Reviews*, Vol. 92, 2018, pp. 658–675. https://doi.org/https://doi.org/10.1016/j.rser.2018.04.106, URL https://www.sciencedirect.com/science/article/pii/S1364032118303241.

[32] Bodkha, "Heat Transfer in Supercritical Fluids: A Review," *Journal of Nuclear Engi-*

neering and Radiation Science, Vol. 7, No. 3, 2021. https://doi.org/10.1115/1.4048898, URL https://doi.org/10.1115/1.4048898, 030802.

[33] TURING, A. M., "I.—COMPUTING MACHINERY AND INTELLIGENCE," *Mind*, Vol. LIX, No. 236, 1950, pp. 433–460. https://doi.org/10.1093/mind/LIX.236.433, URL https://doi.org/10.1093/mind/LIX.236.433.

[34] Chandramouli, S., Dutt, S., Das, A., and Safari, a. O. M. C., *Machine Learning*, Pearson Education India, 2018. URL https://books.google.com/books?id=R6ZJzQEACAAJ.

[35] Sarker, I. H., "Machine learning: Algorithms, real-world applications and research directions," *SN Computer Science*, Vol. 2, No. 3, 2021, pp. 1–21.

[36] Leo, M., Sharma, S., and Maddulety, K., "Machine learning in banking risk management: A literature review," *Risks*, Vol. 7, No. 1, 2019, p. 29.

[37] Lamberton, C., Brigo, D., and Hoy, D., "Impact of Robotics, RPA and AI on the insurance industry: challenges and opportunities," *Journal of Financial Perspectives*, Vol. 4, No. 1, 2017.

[38] Ahmad, M. A., Eckert, C., and Teredesai, A., "Interpretable machine learning in healthcare," *Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics*, 2018, pp. 559–560.

[39] Kwon, B., Ejaz, F., and Hwang, L. K., "Machine learning for heat transfer correlations," *International Communications in Heat and Mass Transfer*, Vol. 116, 2020, p. 104694.

[40] Baghban, A., Kahani, M., Nazari, M. A., Ahmadi, M. H., and Yan, W.-M., "Sensitivity analysis and application of machine learning methods to predict the heat transfer performance of CNT/water nanofluid flows through coils," *International Journal of Heat and Mass Transfer*, Vol. 128, 2019, pp. 825–835.

[41] Lee, D. H., Yoo, J. M., Kim, H. Y., Hong, D. J., Yun, B. J., and Jeong, J. J., "Application of the machine learning technique for the development of a condensation heat transfer model for a passive containment cooling system," *Nuclear Engineering and Technology*, Vol. 54, No. 6, 2022, pp. 2297–2310. https://doi.org/https://doi.org/10.1016/j.net.2021.12.023, URL https://www.sciencedirect.com/science/article/pii/S1738573321006926.

[42] Alizadeh, R., Abad, J. M. N., Ameri, A., Mohebbi, M. R., Mehdizadeh, A., Zhao, D., and Karimi, N., "A machine learning approach to the prediction of transport and thermodynamic processes in multiphysics systems - heat transfer in a hybrid nanofluid flow in porous media," *Journal of the Taiwan Institute of Chemical Engineers*, Vol. 124, 2021, pp. 290–306. https://doi.org/https://doi.org/10.1016/j.jtice.2021.03.043, URL https://www.sciencedirect.com/science/article/pii/S1876107021001516, applications of Nanofluid in Renewable Energy.

[43] McCulloch, W. S., and Pitts, W., "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, Vol. 5, No. 4, 1943, pp. 115–133.

[44] Rosenblatt, F., "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, Vol. 65, No. 6, 1958, p. 386.

[45] Minsky, M. L., and Papert, S. A., "Perceptrons: expanded edition," , 1988.

[46] Werbos, P. J., *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*, Vol. 1, John Wiley & Sons, 1994.

[47] McClelland, J. L., Rumelhart, D. E., and Hinton, G. E., "The appeal of parallel distributed processing," *MIT Press, Cambridge MA*, Vol. 3, 1986, p. 44.

[48] Wu, Y.-c., and Feng, J.-w., "Development and application of artificial neural network," *Wireless Personal Communications*, Vol. 102, No. 2, 2018, pp. 1645–1656.

[49] Sharma, S., Sharma, S., and Athaiya, A., "Activation functions in neural networks," *towards data science*, Vol. 6, No. 12, 2017, pp. 310–316.

[50] Jain, A. K., Mao, J., and Mohiuddin, K. M., "Artificial neural networks: A tutorial," *Computer*, Vol. 29, No. 3, 1996, pp. 31–44.

[51] Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., and Arshad, H., "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, Vol. 4, No. 11, 2018, p. e00938. https://doi.org/https://doi.org/10.1016/j.heliyon.2018.e00938, URL https://www.sciencedirect.com/science/article/pii/S2405844018332067.

[52] Kalogirou, S. A., "Applications of artificial neural-networks for energy systems," *Applied energy*, Vol. 67, No. 1-2, 2000, pp. 17–35.

[53] Baxt, W., "Application of artificial neural networks to clinical medicine," *The Lancet*, Vol. 346, No. 8983, 1995, pp. 1135–1138. https://doi.org/https://doi.org/10.1016/S0140-6736(95)91804-3, URL https://www.sciencedirect.com/science/article/pii/S0140673695918043.

[54] Himmelblau, D. M., "Applications of artificial neural networks in chemical engineering," *Korean journal of chemical engineering*, Vol. 17, No. 4, 2000, pp. 373–392.

[55] Li, H., Zhang, Z., and Liu, Z., "Application of artificial neural networks for catalysis: a review," *Catalysts*, Vol. 7, No. 10, 2017, p. 306.

[56] Huang, Y., Kangas, L. J., and Rasco, B. A., "Applications of Artificial Neural Networks (ANNs) in Food Science," *Critical Reviews in Food Science and Nutrition*, Vol. 47, No. 2, 2007, pp. 113–126. https://doi.org/10.1080/10408390600626453, URL https://doi.org/10.1080/10408390600626453, pMID: 17364697.

[57] Debska, B., and Guzowska-Świder, B., "Application of artificial neural network in food classification," *Analytica Chimica Acta*, Vol. 705, No. 1, 2011, pp. 283–291. https:

//doi.org/https://doi.org/10.1016/j.aca.2011.06.033, URL https://www.sciencedirect.com/science/article/pii/S0003267011008622, a selection of papers presented at the 12th International Conference on Chemometrics in Analytical Chemistry.

[58] Berke, L., and Hajela, P., "Applications of artificial neural nets in structural mechanics," *Shape and Layout Optimization of Structural Systems and Optimality Criteria Methods*, Springer, 1992, pp. 331–348.

[59] Scalabrin, G., and Piazza, L., "Analysis of forced convection heat transfer to supercritical carbon dioxide inside tubes using neural networks," *International Journal of Heat and Mass Transfer*, Vol. 46, No. 7, 2003, pp. 1139–1154. https://doi.org/https://doi.org/10.1016/S0017-9310(02)00382-4, URL https://www.sciencedirect.com/science/article/pii/S0017931002003824.

[60] Olson, D. A., Allen, D. W., et al., "Heat transfer in turbulent supercritical carbon dioxide flowing in a heated horizontal tube," 1998.

[61] Chen, J., Wang, K.-P., and Liang, M.-T., "Predictions of heat transfer coefficients of supercritical carbon dioxide using the overlapped type of local neural network," *International Journal of Heat and Mass Transfer*, Vol. 48, No. 12, 2005, pp. 2483–2492. https://doi.org/https://doi.org/10.1016/j.ijheatmasstransfer.2004.12.040, URL https://www.sciencedirect.com/science/article/pii/S0017931005000943.

[62] Sharifahmadian, A., *Numerical models for submerged breakwaters: coastal hydrodynamics and morphodynamics*, Butterworth-Heinemann, 2015.

[63] Pesteei, S., and Mehrabi, M., "Modeling of convection heat transfer of supercritical carbon dioxide in a vertical tube at low Reynolds numbers using artificial neural network," *International Communications in Heat and Mass Transfer*, Vol. 37, No. 7, 2010, pp. 901–906.

[64] Jiang, R., Wang, X., Cao, S., Zhao, J., and Li, X., "Deep Neural Networks for Channel Estimation in Underwater Acoustic OFDM Systems," *IEEE Access*, Vol. 7, 2019, pp. 23579–23594. https://doi.org/10.1109/ACCESS.2019.2899990.

[65] Chu, X., Chang, W., Pandey, S., Luo, J., Weigand, B., and Laurien, E., "A computationally light data-driven approach for heat transfer and hydraulic characteristics modeling of supercritical fluids: From DNS to DNN," *International Journal of Heat and Mass Transfer*, Vol. 123, 2018, pp. 629–636.

[66] Ye, K., Zhang, Y., Yang, L., Zhao, Y., Li, N., and Xie, C., "Modeling convective heat transfer of supercritical carbon dioxide using an artificial neural network," *Applied Thermal Engineering*, Vol. 150, 2019, pp. 686–695.

[67] Bae, Y.-Y., Kim, H.-Y., and Kang, D.-J., "Forced and mixed convection heat transfer to supercritical CO2 vertically flowing in a uniformly-heated circular tube," *Experimental Thermal and Fluid Science*, Vol. 34, No. 8, 2010, pp. 1295–1308.

[68] Li, Z.-H., Jiang, P.-X., Zhao, C.-R., and Zhang, Y., "Experimental investigation of convection heat transfer of CO2 at supercritical pressures in a vertical circular tube," *Experimental thermal and fluid science*, Vol. 34, No. 8, 2010, pp. 1162–1171.

[69] Kim, D. E., and Kim, M. H., "Experimental study of the effects of flow acceleration and buoyancy on heat transfer in a supercritical fluid flow in a circular tube," *Nuclear Engineering and Design*, Vol. 240, No. 10, 2010, pp. 3336–3349.

[70] Kim, D. E., and Kim, M.-H., "Experimental investigation of heat transfer in vertical upward and downward supercritical CO2 flow in a circular tube," *International Journal of Heat and Fluid Flow*, Vol. 32, No. 1, 2011, pp. 176–191.

[71] Kim, D. E., and Kim, M. H., "Two layer heat transfer model for supercritical fluid flow in a vertical tube," *The Journal of Supercritical Fluids*, Vol. 58, No. 1, 2011, pp. 15–25.

[72] Liu, G., Huang, Y., Wang, J., and Leung, L. H., "Heat transfer of supercritical carbon dioxide flowing in a rectangular circulation loop," *Applied Thermal Engineering*, Vol. 98, 2016, pp. 39–48.

[73] Bae, Y.-Y., Kim, H.-Y., and Yoo, T. H., "Effect of a helical wire on mixed convection heat transfer to carbon dioxide in a vertical circular tube at supercritical pressures," *International journal of heat and fluid flow*, Vol. 32, No. 1, 2011, pp. 340–351.

[74] Jiang, K., "An experimental facility for studying heat transfer in supercritical fluids," Ph.D. thesis, Université d'Ottawa/University of Ottawa, 2015.

[75] Zhu, B., Zhu, X., Xie, J., Xu, J., and Liu, H., "Heat transfer prediction of supercritical carbon dioxide in vertical tube based on artificial neural networks," *Journal of Thermal Science*, Vol. 30, No. 5, 2021, pp. 1751–1767.

[76] Liu, S., Huang, Y., Liu, G., Wang, J., and Leung, L. K., "Improvement of buoyancy and acceleration parameters for forced and mixed convective heat transfer to supercritical fluids flowing in vertical tubes," *International Journal of Heat and Mass Transfer*, Vol. 106, 2017, pp. 1144–1156.

[77] Zhang, Q., Li, H., Kong, X., Liu, J., and Lei, X., "Special heat transfer characteristics of supercritical CO2 flowing in a vertically-upward tube with low mass flux," *International journal of heat and mass transfer*, Vol. 122, 2018, pp. 469–482.

[78] Lei, X., Zhang, J., Gou, L., Zhang, Q., and Li, H., "Experimental study on convection heat transfer of supercritical CO2 in small upward channels," *Energy*, Vol. 176, 2019, pp. 119–130.

[79] Bishop, A. A., Sandberg, R. O., and Tong, L. S., "High-temperature supercritical pressure water loop Part IV: Forced convection heat transfer to water at near-critical temperatures and super-critical pressures," Tech. Rep. WCAP-2056, Westinghouse Electric Corp. Atomic Power Div., Pittsburgh, 1964.

[80] Mokry, S., Pioro, I., and Duffey, R., "Experimental heat transfer to supercritical CO_2 flowing upward in a bare vertical tube," *Supercritical CO2 Power Cycle Symposium*, Troy, New York, 2009.

[81] Yu, J., Jia, B., Wu, D., and Wang, D., "Optimization of heat transfer coefficient correlation at supercritical pressure using genetic algorithms," *Heat and Mass Transfer*, Vol. 45, No. 6, 2009, pp. 757–766. https://doi.org/10.1007/s00231-008-0475-4.

[82] Sun, F., Xie, G., Song, J., Li, S., and Markides, C. N., "Thermal characteristics of in-tube upward supercritical CO2 flows and a new heat transfer prediction model based on artificial neural networks (ANN)," *Applied Thermal Engineering*, Vol. 194, 2021, p. 117067.

[83] Lei, X., Zhang, Q., Zhang, J., and Li, H., "Experimental and numerical investigation of convective heat transfer of supercritical carbon dioxide at low mass fluxes," *Applied Sciences*, Vol. 7, No. 12, 2017, p. 1260.

[84] Song, J., Kim, H., Kim, H., and Bae, Y., "Heat transfer characteristics of a supercritical fluid flow in a vertical pipe," *The Journal of Supercritical Fluids*, Vol. 44, No. 2, 2008, pp. 164–171. https://doi.org/10.1016/j.supflu.2007.11.013.

[85] S. Gupta, I. P., E. Saltanov, "Heat Transfer Correlation for Supercritical Carbon Dioxide Flowing in Vertical Bare Tubes," *Proceedings of the 2013 21st International Conference on Nuclear Engineering*, Chengdu, China, 2013.

[86] Zahlan, H., Groeneveld, D., and Tavoularis, S., "Measurements of convective heat transfer to vertical upward flows of CO2 in circular tubes at near-critical and supercritical pressures," *Nuclear Engineering and Design*, Vol. 289, 2015, pp. 92–107. https://doi.org/10.1016/j.nucengdes.2015.04.013.

[87] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin,

Z., Gimelshein, N., Antiga, L., et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, Vol. 32, 2019.

[88] Jain, A., Patel, H., Nagalapatti, L., Gupta, N., Mehta, S., Guttula, S., Mujumdar, S., Afzal, S., Sharma Mittal, R., and Munigala, V., "Overview and importance of data quality for machine learning tasks," *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 3561–3562.

[89] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, Vol. 12, 2011, pp. 2825–2830.

[90] "Module x2014; PyTorch 1.13 documentation — pytorch.org," https://pytorch.org/docs/stable/generated/torch.nn.Module.html#torch.nn.Module, 2011.

[91] Raju, V. G., Lakshmi, K. P., Jain, V. M., Kalidindi, A., and Padma, V., "Study the influence of normalization/transformation process on the accuracy of supervised classification," *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, IEEE, 2020, pp. 729–735.

[92] "sklearn.preprocessing.RobustScaler — scikit-learn.org," https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html#sklearn.preprocessing.RobustScaler, 2011.

[93] "sklearn.preprocessing.StandardScaler — scikit-learn.org," https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler, 2011.

[94] Ying, X., "An overview of overfitting and its solutions," *Journal of physics: Conference series*, Vol. 1168, IOP Publishing, 2019, p. 022022.

[95] Ruder, S., "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[96] "torch.optim x2014; PyTorch 1.12 documentation — pytorch.org," https://pytorch.org/docs/stable/optim.html, 2011.

[97] Soydaner, D., "A comparison of optimization algorithms for deep learning," *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 34, No. 13, 2020, p. 2052013.

[98] "Adam x2014; PyTorch 1.12 documentation — pytorch.org," https://pytorch.org/docs/stable/generated/torch.optim.Adam.html, 2011.

[99] Loshchilov, I., and Hutter, F., "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

A Appendix Title

Table A.1 Operating Conditions - summary

Author	Equation	Cooling /Heating	Flow Direction	Operating Condition
Bringer & Smith [12]	(2.2)	heating	horizontal	$d = 4.57$ mm, $P = 8.27$ MPa, $Re_b = 3 \cdot 10^4 - 3 \cdot 10^5$ $q = 31.55 - 315.5$ kW/m^2, $T_b = 21 - 49$ °C
Krasnoshchekov & Protopopov [13]	(2.4)	heating	horizontal	$d = 4.08$ mm, $T_b/T_{pc} = 0.9 - 1.2$, $T_w/T_{pc} = 0.9 - 2.5$, $P/P_c = 1.02 - 5.25$, $Re_b = 8 \cdot 10^4 - 5 \cdot 10^5$ $Pr_b = 0.85 - 65$, $\rho_w/\rho_b = 0.09 - 1.0$ $\overline{c}_p/c_{p,b} = 0.02 - 4.0$ $q = 4.6 \cdot 10^4 - 2.6 \cdot 10^6$ W/m^2, $l/d \geq 15$
Petukhov & Kirillov[14]	(2.6)	cooling heating	horizontal	$\mu_w/\mu_b = 0.08 - 40$, $Pr_b = 0.7 - 200$ $Re_b = 10^4 - 10^6$, subcritical
Gnielinski [16]	(2.11)	–	horizontal	$Re_b = 2300 - 10^6$, $Pr_b = 0.6 - 10^5$, subcritical
Pitla et al. [18]	(2.13)	cooling	horizontal	$d = 4.72$ mm, $T_b = 20 - 124$ °C, $\dot{m} = 0.020 - 0.039$ kg/s, $P = 9.4 - 13.4$ MPa
Liao & Zhao [19]	(2.15)	cooling	horizontal	$P = 7.4 - 12$ MPa, $T_b = 20 - 110$ °C $(T_b - T_w) = 2 - 30$ °C, $\dot{m} = 0.02 - 0.2$ kg/min $Ri_b = 10^{-5} - 10^{-2}$, $d = 0.5 - 2.16$ mm
Dang & Hihara [2]	(2.21)	cooling	horizontal	$d = 1 - 6$ mm, $l = 500$ mm, $P = 8 - 10$ MPa, $T_{b,in,CO_2} = 30 - 70$ °C, $G_{CO_2} = 200 - 1200$ kg/m^2·s, $q = 6 - 33$ kW/m^2

Table A.2 Input and output parameters used to predict sCO_2 thermal behavior.

Author		Output	Input	Operating Range
Scalabrin & Piazza [59]	1.)	Nu	Re, Pr, Ec	–
Chen et al. [61]	2.)	α	P_r, T_r, $\dot{m}$, q	
	3.)	Nu	Re, Pr, $\frac{\rho_w}{\rho_b}$, $\frac{\overline{c}_p}{c_{p,b}}$	
	4.)	α	P_r, T_r, $\dot{m}$, $\frac{T_w}{T_b}$	
Pesteei & Mehrabi [63]		α_x	Re, G, Bo^*, x^+, q	$q = 4.49 - 36.8$ kW/m^2, $Re_{in} = 1810 - 1993$, $T_{b,in} = 24.6$ °C, and $P = 9.57$ MPa
Chu et al. [65]		T_w, τ_w	d, P, T_{in}, h_b, q	$d = 2, 5, 10$ mm, $q = 5, 10, 20, 30$ kW/m^2, $T_{in} = 15, 28$ °C, $P = 8, 8.8$ MPa
Ye et al. [66]		T_w	d, P, G, h_b, q	$d = 2 - 22$ mm, $T_b = -6 - 115$ °C, $P = 7.5 - 9.23$ MPa, $G = 100 - 3079$ kg/m^2·s, and $q = 0.479 - 616.3$ kW/m^2
Zhu et al. [75]				$d = 2 - 16$ mm, $P = 7.5 - 20.8$ MPa, $q = 5 - 350$ kW/m^2, and $G = 488 - 2000$ kg/m^2·s

A.1 Code

The following code can be used to upload the final model, scaling parameters and model definition. This code has two parts, one where a range of Bulk temperaturess in .xlsx file can be used as input and the second part where a single bulk temperature point can be used as an input.

A.1.1 Part 1 - Range of bulk temperature values

The data science libraries are imported here

#required libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import torch
import torch.nn as nn
from torchsummary import summary
from google.colab import files
```

The google drive is connected so that all the data files can be accessed/uploaded directly from/to the drive

```
#mounting the google drive
from google.colab import drive
drive.mount('/content/drive')

#upload bulk temperature file (xlsx file with a single column)
uploaded = files.upload()

#enter the heat flux and Mass flux value
q = 15
G = 550
Input = pd.read_excel("Bulk_Temperature.xlsx")
Tb_new = np.array([Input]).reshape(-1,1)
q_new = q*np.ones([len(Tb _new),1])
G_new = G*np.ones([len(Tb _new),1])
X_test_new = np.concatenate((Tb_new,q_new,G_new),axis = 1)

#Import ScalerX and ScalerY files
```

```
from pickle import load
scaler _x = load(open('/content/drive/MyDrive/final _model _scalerX.pkl', 'rb'))
scaler _y = load(open('/content/drive/MyDrive/final _model _scalerY.pkl', 'rb'))

#Import the Model _def
import sys
sys.path.append("/content/drive/MyDrive/Model_def") location where Model_def folder is
located (Note: Enter file name in the next line)
from ModelDef import * # (f3 is the file name)

#Import the Model
model = sCO2()
PATH_model = '/content/drive/MyDrive/final_model.pt' load the model path here
new_model = torch.load(PATH_model)

# Predicting

new_model.eval()
X_test_norm = scaler_x.transform(X_test_new)
X_test_norm_t = torch.from_numpy(X_test_norm.astype(np.float32))
with torch.no_grad():
Yhat_norm = new _model(X _test _norm _t).numpy()
pred_Y = scaler _y.inverse _transform(Yhat _norm)

print(pred_Y)
```

A.1.2 Part 2 - Single point prediction

#required libraries

import numpy as np

import torch

Input values

Tb = 299 #*Enter bulk temperature in K*

q = 6 #*Enter Heat flux in* kW/m^2

G = 1000 #*Enter Mass flux* $kg/m^2.s$

inputs = np.array([Tb, q, G]).reshape(1,-1)

mounting my google drive

from google.colab import drive

drive.mount('/content/drive')

#Import ScalerX and ScalerY files

from pickle import load

scaler _x = load(open('/content/drive/MyDrive/final _model _scalerX.pkl', 'rb'))

scaler _y = load(open('/content/drive/MyDrive/final _model _scalerY.pkl', 'rb'))

#Import the Model _def

import sys

sys.path.append("/content/drive/MyDrive/Model_def") location where Model_def folder is located (Note: Enter file name in the next line)

from ModelDef import * # (f3 is the file name)

#Import the Model

```
model = sCO2()
PATH_model = '/content/drive/MyDrive/final_model.pt' load the model path here
new_model = torch.load(PATH_model)

# Predicting

new_model.eval()
X_test_norm = scaler_x.transform(X_test_new)
X_test_norm_t = torch.from_numpy(X_test_norm.astype(np.float32))
with torch.no_grad():
Yhat_norm = new _model(X _test _norm _t).numpy()
pred_Y = scaler _y.inverse _transform(Yhat _norm)

print(pred_Y)
```