University of South-Eastern Norway

www.usn.no

FMH606 Master's Thesis 2023

Industrial IT and Automation

Velocity Monitoring of Phases in Oil/Gas/Water Multiphase Flow

Abdur Rahman

Faculty of Technology, Natural sciences and Maritime Sciences

Campus Porsgrunn

Course: FMH606 Master's Thesis, 2023

Title: Velocity monitoring of phases in oil/gas/water multiphase flow

Number of pages: 90

Keywords: Data denoising, flow rate prediction, neural networks, flow type classification, physics-based flow rate prediction

Student: Abdur Rahman

Supervisors: Ru Yan

Saba Mylvaganam

External partners: Kjetil Fjalestad (EQUINOR)

Tonni Franke Johansen (SINTEF)

Summary:

Machine learning and deep learning techniques have gained significant attention in recent years for enhancing the precision and efficiency of velocity estimation in multiphase flow. This thesis aims to achieve three objectives: predicting the flow rate of multiphase flow using only accelerometer data from four installed accelerometers in Equinor's test rig, creating spectrograms from accelerometer data and then train a CNN (Convolutional Neural Network) model for flow type identification, and exploring the possibilities of applying physics-based machine learning in case of flow rate predictions of multiphase flow using only accelerometer data.

The results are promising in predicting flow rate and classifying flow types using machine learning techniques. However, the denoising process applied was assumed to be effective, and a more reliable and accurate denoising process for filtering out the noise in the accelerometer data caused by the installed Coriolis meter is necessary for future studies. The study has demonstrated the potential to predict flow rate using fewer than all four accelerometers. However, the models' proficiency is linked to the number of samples available for training. It is worth noting that more samples are required for single-phase flows to further improve the accuracy of the model.

The thesis has successfully demonstrated that accelerometer data from Equinor test rig contains the information to predict multiphase flow rate and to identify flow types of multiphase flow utilizing machine learning techniques. Moreover, the study has shown the potential use of physics-based machine learning in the case of making predictions of multiphase flow rate.

`

Preface

The report is written by Abdur Rahman as a student of Master of Science in Industrial IT and Automation as master's thesis at University of South-Eastern Norway, from January to May in 2023. This thesis involves data analysis, denoising, multiphase flow rate prediction, flow type classification and application of physics in machine learning to predict flow rate. The necessary data has been provided by Equinor. This work is closely coupled to an ongoing SAM project (Self Adapting Model-based system for Process Autonomy - SINTEF)

It is assumed that the reader has previous knowledge in signal processing, machine learning/ deep learning and image recognition.

First and foremost, I would like to thank the Almighty, the most Merciful, the most Gracious, the Wisest and the most Knowledgeable. I thank my supervisors Ru Yan and Saba Mylvaganam for their intense support and guidance during the entire thesis work. I would also like to thank the external partners for their support during the thesis. I am eternally indebted to my parents for their support and encouragement throughout my career. I thank my family, friends and loved ones.

The following software and frameworks were used during the thesis work:

- Microsoft office 365
- Python 3.9.16
- JupyterLab
- TensorFlow 2.8.2
- Scikit-learn 1.2.2
- SciPy 1.10.1

Porsgrunn, 15th May 2023

Abdur Rahman

Contents

Nomenclature

ANN – Artificial Neural Network

CNN – Convolutional Neural Network

CFD – Computational Fluid Dynamics

DNN – Deep Neural Network

FFT – Fast Fourier Transform

GBDT - Gradient Boosting Decision Tree

PBML – Physics-based Machine Learning

ReLU – Rectified Linear Unit

RMSE – Root Mean Square Error

SVM – Support Vector Machine

STFT – Short Term Fourier Transform

List of symbols

f_d	Darcy friction factor
L	Length of the pipe or duct.
D	Diameter of the pipe or duct.
v	Velocity of the fluid.
g	Acceleration due to gravity.

1 Introduction

Multiphase flow, a phenomenon where two or more phases of matter, such as oil, gas, and water, flowing simultaneously, is ubiquitous in the oil and gas industry [1]. Accurate monitoring and prediction of each phase velocity in these flows is essential for various purposes, including production system design, optimization, flow pattern prediction, and identification of flow regimes. The magnitude and complexities involved in multiphase flow necessitate accurate predictions of fluid phase velocities, which can only be accomplished through sophisticated monitoring techniques.

In recent years, there has been growing interest in using machine learning and deep learning techniques to improve the accuracy and efficiency of velocity estimation in multiphase flow [2]. These techniques have the potential to provide insights into the complex physical phenomena that occur during multiphase flow and enable the development of more accurate predictive models.

The objective of this thesis is to investigate machine learning and deep learning techniques for monitoring and predicting phase velocities in oil/gas/water multiphase flow using data from accelerometers. Data from Equinor's multiphase flow rig in Porsgrunn, Norway, will be used to train and assess the performance of these models. The focus will be on developing data driven models capable of estimating the velocities of all three phases simultaneously based on the sensor input data. To identify flow types, image recognition techniques are to be implemented utilizing machine learning and deep learning techniques. Furthermore, physics-based machine learning methods are to be explored, integrating sensor data to estimate flow velocities.

The data used to develop models in this study was collected from tests conducted at Equinor's multiphase flow rig, using natural gas, crude oil, and water. The rig has a three-inch inner diameter flow loop, and the temperature and mass flow varied during the tests. Sensors, including accelerometers, differential pressure, and pressure sensors, were placed in specific locations within the rig (as shown in Figure 1).

Figure 1 displays a simplified diagram of Equinor's test rig with the positions of the four accelerometers to capture vibration readings from the rig surface.

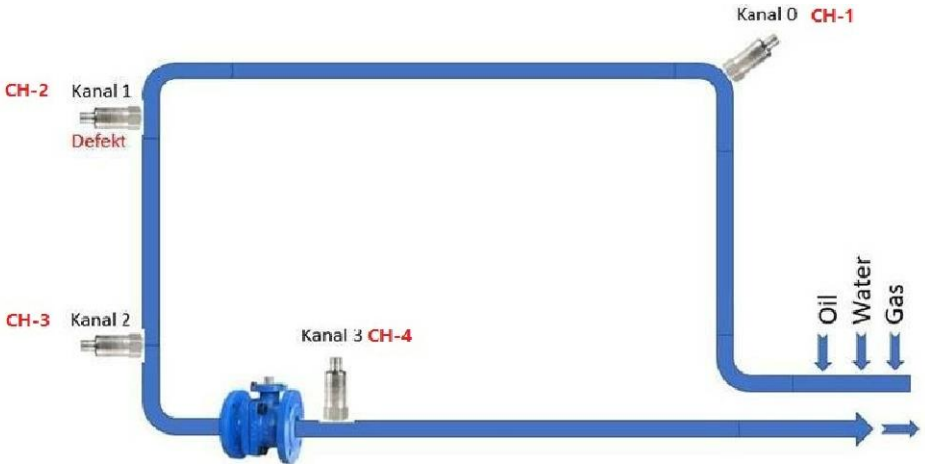

Figure 1: Simplified diagram of Equinor test rig showing only the accelerometers (Acoustic emission sensors),[3]

This research has potential to advance the understanding and prediction of multiphase flow and improve the efficiency and effectiveness of oil and gas production using machine learning techniques.

A signed copy of the task description of the thesis can be found in Appendix A.

The report is structured in the following sequence:

- Chapter 1 provides an introduction and objectives of the thesis, followed by a brief background review of the previous works relevant to multiphase flow velocity monitoring and machine learning techniques.
- Chapter 2. This chapter contains some brief theoretical descriptions that are relevant to this thesis.
- Chapter 3 describes the data denoising and preparation process.
- Chapter 4 will describe the implementation of a data-driven flow velocity estimator model.
- Chapter 5 covers the flow type identification model implementation with deep learning image recognition technique.
- Chapter 6 illustrates the use of physics-based machine learning for estimation of flow velocity of multiphase flow.
- Chapter 7 shows the results obtained from the data driven model, flow type identification model and physics-based model. It covers brief discussion on the performance of the models as well.
- Chapter 8 covers discussion of the work and results. It also discusses the issues with the data that were faced during the thesis.
- Chapter 9 concludes the thesis report and suggests some possible future work on the relevant subjects.

1.1 Background

The realm of fluid mechanics has long been involved in the quest for measurement and prediction of multiphase flow, a critical research area across various industries, including chemical engineering, oil and gas, and power generation. The intricate nature of flow and the presence of diverse phases pose a challenge to researchers, who have utilized various methodologies such as computational fluid dynamics (CFD) simulations, experimental methods, and empirical modeling.

Empirical approaches rely on multiphase flowmeters to evaluate multiphase flows, providing direct measurements of phase proportions and flow rates, [2], [4]–[6]. However, these techniques can be costly, complicated, and unsuitable for every multiphase flow scenario. Conversely, CFD prognoses have demonstrated potential in estimating multiphase flows but require high computational resources and may not capture the elaborate physics of flow.

Recently, artificial intelligence (AI) methods have been suggested as a potential resolution for multiphase flow measurement and prediction. AI models were employed for multi-phase flow rate prediction and flow regime identification using data-driven methodologies, [2]–[4], [7]. In these investigations, models are trained on empirical data and have demonstrated to offer precise prediction of multiphase flow rate and flow categories. Additionally, image recognition techniques utilizing Convolutional Neural Networks (CNNs) were implemented to categorize the flow patterns of multiphase flows. These models have been trained with experimental images and have been confirmed effective in identifying distinct flow patterns in multiphase flows.

Empirical models such as the drift flow model and the smooth flow model have been developed to predict multiphase flows, [8], [9]. These models are based on experimental data and are easy to implement, but they are often not accurate for all types of multiphase flows.

In the paper “Ensemble learning in the estimation of flow types and velocities of individual phases in multiphase flow using non-intrusive accelerometers’ and process pressure data”[3], the test data from a previous experiment using Equinor’s test rig were analysed. This study presents a new method for accurately identifying different types of flow and estimating the velocity of each phase in a multiphase fluid system. The proposed method uses both pressure measurements from differential transmitters and pipe vibrations caused by the flow, making it more precise than traditional methods such as ultrasonic sensors or gamma ray densitometers. Machine learning models based on decision trees have been developed using data collected from accelerometers, differential pressure transmitters, and upstream/downstream pressure transmitters, to identify the ratio of different phases (such as water cut) and estimate the velocity of each phase, as well as the status of choke valves if present. To comprehensively evaluate the replicability and versatility of the model, further research must be conducted, encompassing a broader range of phase fractions and flow rates. Moreover, increasing the frequency of data logging for pressure measurements would allow for a deeper understanding of the effects of discrete process data on the results. To enhance the understanding of how discrete process data (i.e., pressure measurements) may influence the results, elevating the

frequency of data logging for pressure measurements would permit a more profound comprehension of the ramifications of discrete process data on the outcomes.

Research conducted at The University of Edinburgh named “Comparison of machine learning methods for multiphase flowrate prediction” studied the estimation of instantaneous flowrate of gas/oil/water three-phase flows using Deep Neural Network (DNN), Support Vector Machine (SVM) and Gradient Boosting Decision Tree models (GBDT)[2] This research aimed to determine the instantaneous flow rate of a three-phase flow of gas, oil, and water by using a combination of a Venturi tube and various machine learning techniques. The results of the experiments showed that both DNNs and SVM were able to provide accurate estimates for multiphase flows, while the GBDT model was not successful in this task. It was discovered that the volumetric gas phase flow rate could be accurately predicted using a SVM model, and temperature was found to be an important factor when estimating multiphase flows. Further research should be conducted to assess performance of DNN, SVM and GBDT models’ effectiveness in real world applications with different types of data sets. Additionally, more data pre-processing operations are needed to be explored to improve the accuracy and reliability when estimating multiphase flows using these machine learning methods.

In recent times, physics-based machine learning methods have been proposed to create multiphase models. These models incorporate the governing equations of fluid dynamics. Physics aided machine learning was utilized to estimate flow rate of a multiphase flow system in research named “Machine Learning and First Principles Modeling Applied to Multiphase Flow Estimation” by Timur Bikmukhametov in 2020, [10]. This study presents a robust and accurate method for estimating multiphase flow by combining machine learning models with fundamental principles of physics. The authors were able to tune the model accurately to match specific field conditions and estimate the uncertainty of their predictions based on the process conditions. Additionally, this work demonstrates how using features based on physical laws can lead to improved prediction performance compared to using raw data alone. The proposed method is both accurate and interpretable, providing useful insights for practitioners. However, the paper assumes that measurements are free from noise and errors, which is not always the case, as random and possibly drift errors may always present, making implementation more challenging.

In multiphase flow research, the Darcy-Weisbach equation is widely used to model the pressure drop in pipe flow systems. The equation is often used in combination with other models and methods to provide more accurate predictions of multiphase flow[11], [12]. Research named “Two-phase flow pressure drop in PEM fuel cell flow channel bends” by Mehdi Mortazavi, in which the two-phase flow pressure drop across a PEM fuel cell flow channel bend is investigated. In this study, the Darcy-Weisbach equation is used to calculate the pressure drop across pipes and other flow channels, [13].

In recent years, researchers have also been using machine learning techniques to model multiphase flow and the Darcy-Weisbach equation is used to create physics-based machine learning models. In a paper named “Machine Learning for Closure Models in Multiphase-Flow

Applications", Darcy-Weisbach equation has been utilized to model the closure terms in two-fluid models, [14].

This thesis focuses on developing a data-driven deep learning model to estimate flow rate of multiphase flow. Furthermore, an image recognition CNN model is to be developed to identify flow types in the multiphase flow system and explore a physics-based machine learning model to predict single phase flow rate with utilizing the Darcy-Weisbach equation.

The measurement and prediction of multiphase flow rate have been the subject of intensive research for several years. Despite the availability of experimental methods, CFD simulations, and empirical models, the accuracy and effectiveness of these techniques are limited. As such, the academic community has shifted its focus towards machine learning frameworks as a plausible remedy for gauging and prophesying multiphase current. By utilizing the capabilities of synthetic cognition, these frameworks have the possibility of transforming the domain of multiphase current quantification and prophecy. These models use data-driven approaches and have been shown to provide accurate predictions of multiphase flow rate. Image recognition using CNNs have been applied for classifying the flow patterns of multiphase flows, and physics-based machine learning models have been proposed for modelling multiphase flow systems.

2 Theory

In this chapter, the theoretical concepts are briefly discussed to provide context for research presented in this thesis.

2.1 Multiphase flow

Multiphase flow is an intricate phenomenon that involves the simultaneous flow of different materials like gas, liquid, and solid through a pipe or conduit. The complexity arises due to the intricate interplay between these different phases, which makes predicting and controlling their behavior a challenging task, [1]. The flow of these different phases is influenced by numerous factors, including flow rates, pressure, phase distribution, and flow regime. The flow regime describes how these different phases are arranged and interact with each other. For instance, stratified flow refers to the formation of layers, while dispersed flow describes when the different phases are mixed up. Factors such as the pipe or conduit geometry, density, and viscosity of the phases, and their flow rates influence the flow regime and phase distribution. Studying multiphase flow is indeed a challenging and intricate field, but the use of sophisticated models and techniques makes it possible to predict and control the flow.

2.2 Accelerometer

In the field of multiphase flow research, accelerometers are a crucial tool for understanding and analyzing the complex behavior of the flow. They allow researchers to non-intrusively measure the flow conditions in pipelines [15]. These devices are used to measure the vibrations caused by the flow of different phases through a pipeline, allowing researchers to gather valuable information about the flow rate and flow type. In essence, accelerometers detect changes in velocity over time and convert them into electrical signals. This is achieved using a proof mass, which is a small mass that is suspended within the accelerometer. When a device is subjected to acceleration, the proof mass moves in response, resulting in the compression or stretching of a spring.

Accelerometers can be classified into several types, such as piezoelectric, capacitive, and piezoresistive accelerometers, [16]. Piezoelectric accelerometers are most used in multiphase flow research, as they are rugged and can operate in harsh environments. They also have a high sensitivity and can measure a wide range of frequencies, which makes them well suited for measuring vibrations caused by multiphase flows. Nevertheless, they are also vulnerable to noise generated by other sources, such as the Coriolis meter, which is frequently used in multiphase flow research. Consequently, it is vital to denoise the accelerometer data to obtain precise results.

2.3 Artificial neural networks (ANNs)

The innovative artificial neural networks (ANNs) are machine learning models that get their inspiration from the biological neural networks' structure and function. They consist of interconnected "neurons" that proficiently process and transmit data. ANNs have widespread usage in various fields, such as image recognition, natural language processing, and prediction tasks. Specifically, in the multiphase flow context, ANNs have been employed to predict flow patterns, estimate discharge, and classify flow regimes [17], [18]. Their efficacy lies in their ability to handle nonlinear and complex relationships in multiphase flow data. ANNs have been hailed for their unmatched capacity to handle vast amounts of data and adapt to ever-changing conditions, making them an essential tool for the prediction of multiphase flows.

2.4 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a formidable type of deep learning architecture, expertly crafted to excel in tasks such as image recognition and classification with exceptional precision. Comprising a sophisticated network of interconnected "neurons," CNNs collaboratively process and manipulate input data to ultimately generate a set of output predictions. To unravel the complexities of patterns and features embedded within input data, CNNs utilize a technique called convolution, [19], [20]. This process involves deploying a set of small filters to scrutinize different segments of the input image. These filters glide over the image, seamlessly extracting information at various scales.

In the realm of multiphase flow type detection, CNNs have shown remarkable efficacy in image recognition tasks and have demonstrated great potential for analyzing time series data. The primary concept behind utilizing CNNs for this purpose is to convert raw accelerometer data into images, which can then be analyzed by the CNN to identify patterns and features corresponding to various flow types [20]–[22]. One of the most significant benefits of using CNNs for multiphase flow type detection is that they can automatically extract features from raw data, eliminating the requirement for manual feature engineering. This attribute is particularly advantageous when dealing with complicated and intricate multiphase flow data that may be difficult to interpret. Additionally, CNNs can process high-dimensional data, which is frequently encountered in multiphase flow measurements.

2.5 Coriolis meter

A Coriolis meter is a type of flow meter that is commonly used in the oil and gas industry to measure the mass flow rate of liquids and gases. The Coriolis meter works by measuring the Coriolis effect, which is a phenomenon that occurs when a fluid flows through a tube that is vibrating at a specific frequency. The Coriolis effect causes a slight bending of the tube, which can be measured and used to calculate the flow rate of the fluid [23], [24]. This measurement procedure of Coriolis meter creates vibration in the flowing fluid. As accelerometers capture

vibrations, the vibration created by Coriolis meter adds noise to accelerometers' measurements. Thus, filtering the accelerometer data becomes necessary.

2.6 Short-time Fourier Transform (STFT)

STFT is an essential and mighty tool utilized in the domain of signal processing and analysis, particularly in the field of audio and vibration analysis [25]. Its operating mechanism involves breaking down a signal into overlapping segments and executing a Fourier Transform on each chunk, resulting in a comprehensive analysis of the signal in the time-frequency domain.

When compared to other common time-frequency analysis methods such as the Fourier Transform and Wavelet Transform, STFT possesses several benefits. Specifically, it can provide an exceptional temporal resolution for high-frequency signal components while concurrently providing a high-frequency resolution for low-frequency components [25]. Therefore, STFT can offer a precise and accurate depiction of the frequency content of a signal at any moment in time.

The remarkable capabilities of STFT in terms of its temporal and frequency resolution make it an exceptionally valuable tool for vibration analysis and many other fields [25], [26]. It can quickly and accurately detect and isolate unwanted noise and extract critical information about the signal of interest, thereby enabling researchers and engineers to make more informed and reliable decisions.

2.7 Physics-based machine learning

The rapidly growing discipline of physics-based machine learning (PBML) endeavors to combine the fundamental physical principles of a given system with machine learning models to enhance their precision and comprehensibility. This strategy focuses on the integration of domain-specific knowledge regarding physical processes into the models, ultimately yielding more nuanced and insightful predictions [27], [28]. Diverging from traditional machine learning techniques that exclusively rely on data for model training, PBML employs physical principles such as equations that describe the physical behavior of the system under examination. By doing so, PBML produces more robust and accurate predictions and provides valuable insights into the underlying physical processes driving the data. Physics-based machine learning has proved to be highly useful in various fields, including fluid dynamics, materials science, and climate modeling.

3 Data Preprocessing and Feature Extraction

Only the accelerometer data has been utilized in this thesis from the data provided by Equinor. The source and preparation of the data is briefly discussed in this chapter below:

3.1 Accelerometer data source

The description of data source below is taken from previously conducted research on the same data named “Ensemble learning in the estimation of flow types and velocities of individual phases in multiphase flow using non-intrusive accelerometers and process pressure data”.[3] The data was collected from Equinor's multiphase flow test rig in Porsgrunn, Norway. It was gathered using accelerometers (piezoelectric transducers that measure acceleration), differential pressure transmitters and upstream-and downstream pressure transmitters. The diameter of the test rig in the tests is 3 inches, and the materials that have been used were natural gas, crude oil and water. The temperatures readings on the upstream and downstream sides of the flow varied between 65°C and 94°C and the total mass flow rate varied from approximately 1 ton per hour to 64 tons per hour. The test setup at the rig has been shown in a simplified diagram in Figure 1, which shows the flow direction, locations of choke valves, and the sensors and instruments used to collect data. The four explosion-proof piezoelectric accelerometers were placed along the flow direction, with two at the second and fourth bends, and one close to the choke valve at the outlet side. Differential pressure measurements were obtained from two Venturi meters located in the second riser section and the last flow loop section. The accelerometer sampling frequency was set at 50 kHz, and acceleration data were logged continuously for 5 minutes before the next series started. Thus, there are about 15 million data points in one acceleration measurement from each test. Other measurements, such as differential pressure and upstream and downstream pressure data, were logged as discrete process data, and only their average values were saved. Therefore, except for the data from the four accelerometers, all the other measurements have only a single value per test in the datasets.

3.2 Denoising data

The accelerometers attached to the rig record vibrations from the rig walls. The thesis works on the data that is acquired from four accelerometers installed on the test rig. However, A Coriolis meter is already installed at the very beginning of the rig as well to measure flow rates.

Coriolis flowmeter can add noise to the accelerometer data due to its measuring mechanism. Noise is added to accelerometer data from Coriolis meter because by design, the accelerometers measure vibrations caused by the flow in the rig, whereas the Coriolis meter is designed to measure vibrations created by the flow in the meter itself. The vibrations caused by the Coriolis meter can interfere with the accelerometer data and make it difficult to precisely record the

flow in the rig. To overcome this problem, it is important to denoise the accelerometer data from the Coriolis meter. This can be done by using different signal processing techniques such as filtering and signal separation algorithms. Denoising the accelerometer data may improve the accuracy of the flow rate measurements and provide more reliable results.

Short-time Fourier Transform (STFT) has been used to analyze accelerometer data to remove supposed unwanted noise and extract useful information about the frequency content of the signal. The overlapping segments of the signal were analyzed using a Fourier Transform. The supposed unwanted noise in the frequency range of 190-210 Hz and its harmonics, first harmonics 380-420 Hz and second harmonics 760-840 Hz were identified and removed from the signal using a masking technique.

In Figure 2, frequency component plot has been shown to analyse the difference of a sample before and after filtering. Some clear distinction can be observed at around 200 Hz, 400 Hz and 800 Hz as these are the regions where masking has been applied. Observing the colour bars; it can be noticed that some low amplitude data has been removed from the original data.

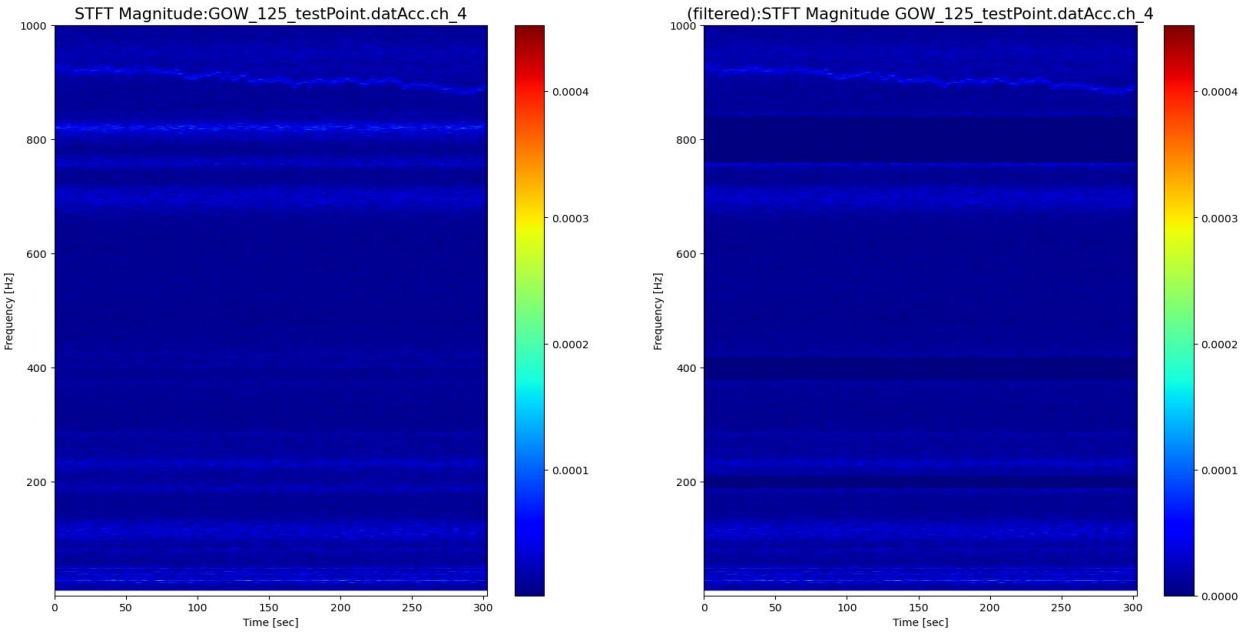

Figure 2: An example of Frequency component plot using STFT – before and after applying masking for supposed denoising.

Figure 3 has original and filtered data plotted on top of each other. Very little difference can be observed in this plot at the edges. Original data seems to be a little bit higher in magnitude or it contains more data.

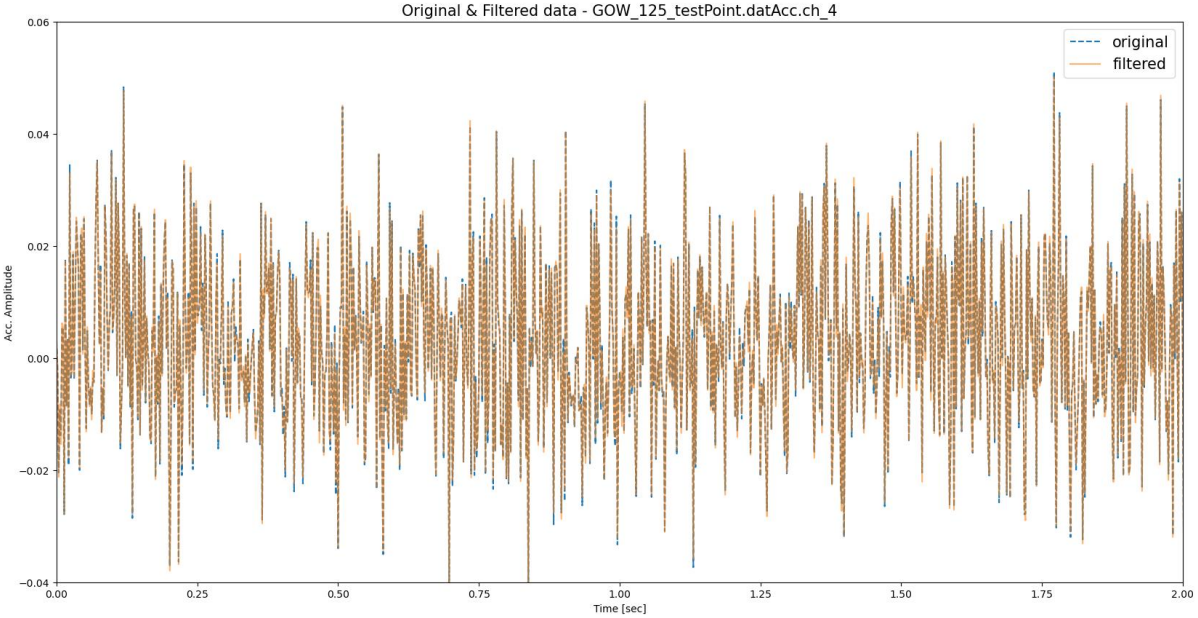

Figure 3: An example of original data and filtered data.

In Figure 4a comparison between original accelerometer data and the supposed noise due to Coriolis meter has been shown. It should be noted that it is not certain that this noise the true noise due to Coriolis meter. The data analysts at Equinor informed that Coriolis meters attached into the rig add low amplitude noise of around 200 Hz.

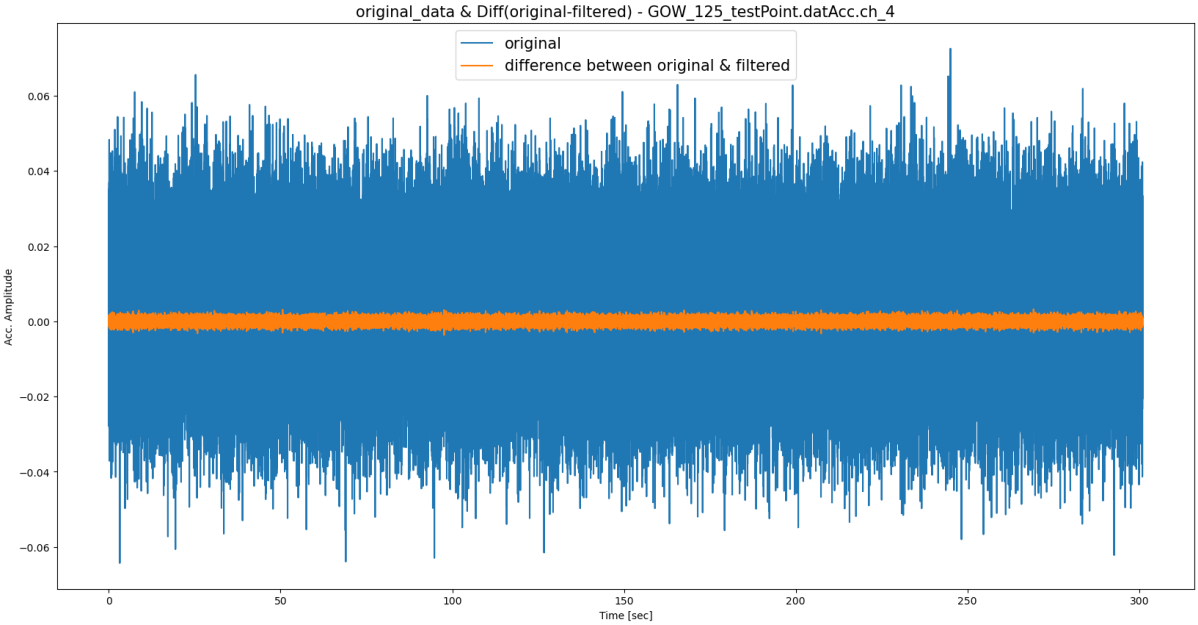

Figure 4: An example of comparison between original data and noise.

From Figure 5, difference between original data and filtered data can be observed due to filtering process within the range of 0 Hz to 1000 Hz.

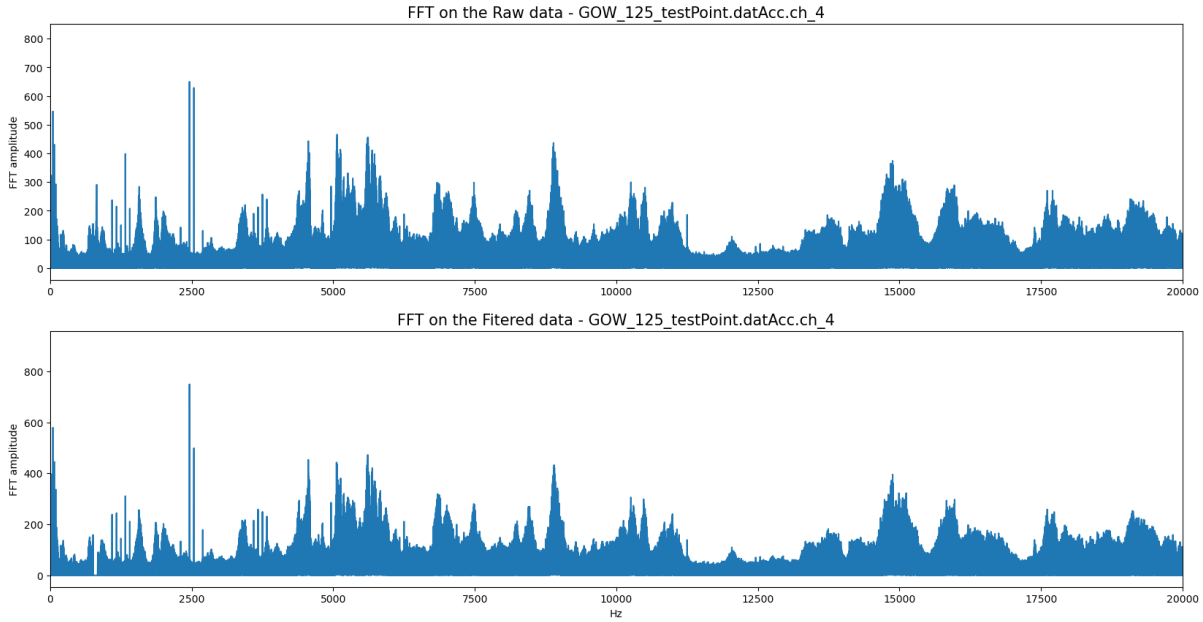

Figure 5: An example of FFT plot – raw/original data and filtered data.

Figure 6 shows the difference between the original data and filtered data. The difference is considered as the Coriolis noise that has been filtered out by STFT and masking.

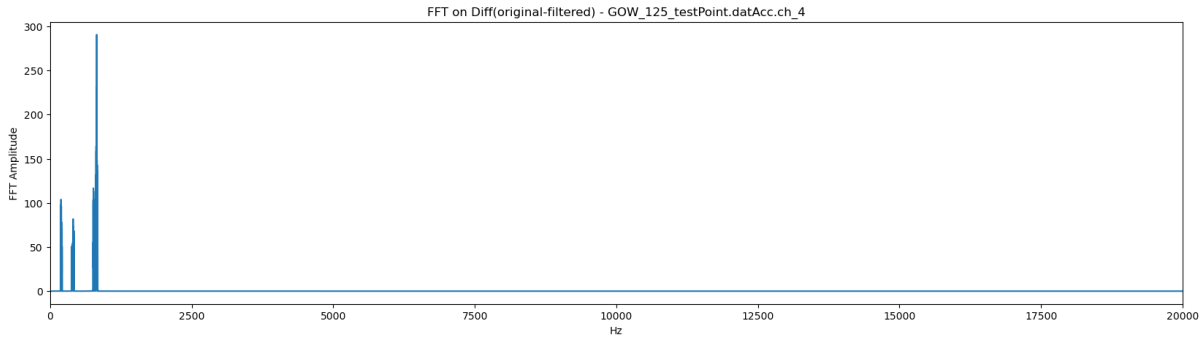

Figure 6: An example of Extracted supposed noise.

Observing Figure 2 and Figure 4 it is understood that the data removed from original data seems to somewhat match the expected characteristic of Coriolis noise as the removed data is of low amplitude and exists within the defined frequency range including its first and second harmonics.

3.3 Peaks extraction from accelerometer data

In the available dataset, there are 217 usable samples recorded by the four accelerometers. Per sample, the measurement from each accelerometer contains 15411200 datapoints, with a sampling duration of 301 seconds and a sampling interval of 0.000019531 second. To utilize this large amount of data in a machine learning model to predict flow rates of gas, oil, and water; impactful and important information has been planned to be extracted from raw data instead of using all the data.

One of the crucial information that is being extracted from the raw data is peak amplitudes and frequencies. Peaks represent the specific frequencies at which certain components of the signal have the highest energy. Extraction of peaks is a common technique used in signal processing to identify specific events or features within a signal, [29]. The peaks in the frequency domain have been identified using the 'signal.find_peaks' function from 'SciPy' library. The function has been set to detect peaks with given minimum height and distance between the peaks.

Figure 7 illustrates the detected peaks in the FFT of a sample, marked with a cross (x).

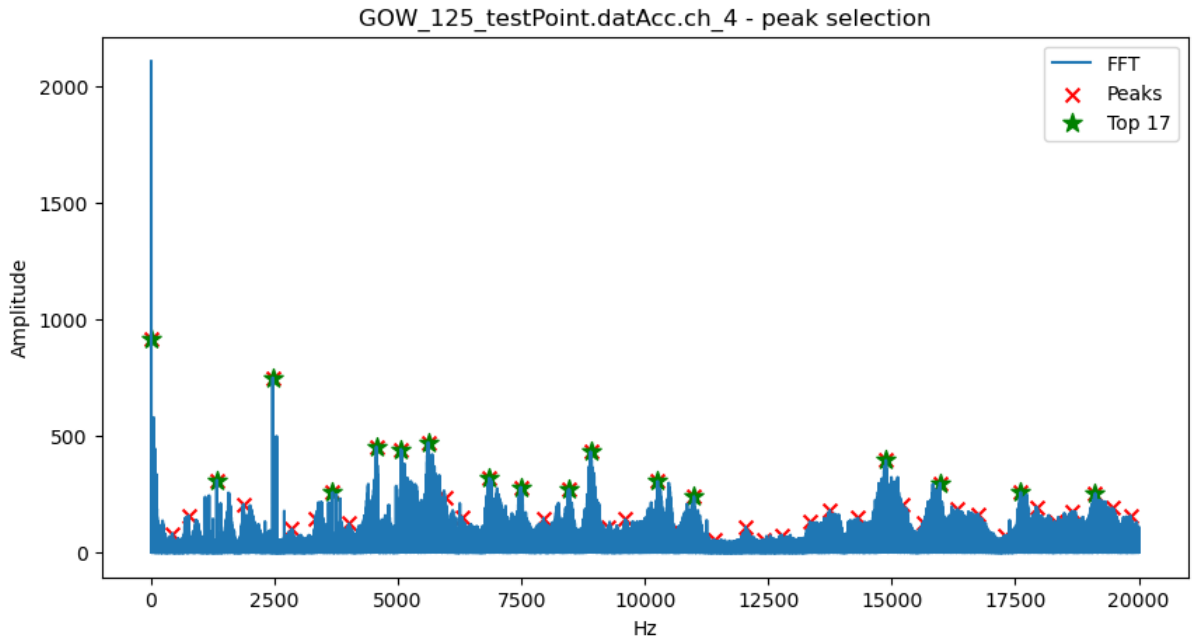

Figure 7: An example of Peak detection and top 17 peaks selection.

After identifying the peaks, the amplitudes and frequencies of the peaks have been obtained and sorted in descending order based on the amplitudes. The top 17 amplitudes and their corresponding frequencies have been selected as the features to be used in the machine learning model. These top 17 selected peaks are clearly marked by green stars (⋆) in the Figure 7, as an example. The top 17 features have been chosen based on the observation that they contain the most significant information that is relevant to the prediction of flow rates of gas, oil and water, as the selected peaks are denoted by green stars (⋆).

3.4 Automated script for peaks/feature extraction from accelerometer data

To optimize the process of extracting valuable information from raw data, an automated Python script has been devised. The script carries out the complete procedure by accepting the raw data as an input and then undertakes several crucial stages to extract the requisite information. These steps include filtering the data, conducting the Fourier transform, identifying, and extracting peaks, and applying a binary mask to get rid of noise and harmonics. Upon the completion of these steps, the script then extracts the peak values and deposits them into a csv file along with their corresponding test ID. The modular and adaptable design of the script ensures that it can be customized and modified to suit any specific requirements.

The automated extraction of significant information from raw data streamlines the analysis process and saves a considerable amount of time. Moreover, the script ensures that the data is processed with utmost consistency and accuracy, effectively minimizing the chances of human error. The script's implementation involves running a loop that accepts all 217 samples as inputs and generates 17 peak values for each sample.

3.4.1 Explanation of code for extracting features

Figure 8 illustrates a flow diagram of the feature extraction script that is used to preprocess and extract features from the dataset that will be used to train a machine learning model to predict flow rates of multiphase flow.

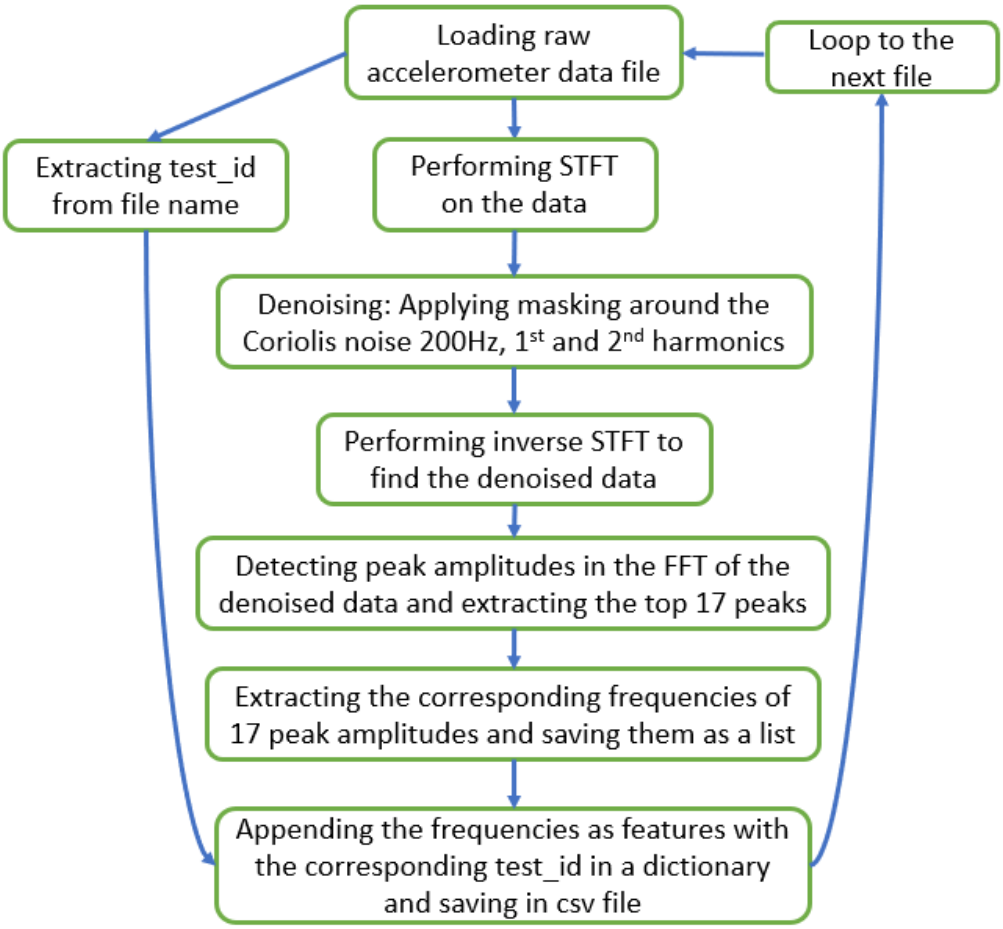

Figure 8: Flow diagram of feature extraction code from accelerometer data.

The script loads csv a file in specific directory, test_id from the file name of the data is saved separately. The script performs STFT on the data, then applies masking within the defined ranges, then performs inverse STFT to gain the denoised data. Then FFT is performed on the denoised data and peak amplitudes are detected. The corresponding frequencies of the top 17 peak amplitudes are taken and kept as feature data. The 17 frequencies with the test ID are saved in a csv file. Then the script moves on the next file in the directory. The feature extraction code is available in Appendix B for further reference. The code in Appendix B contains only the main process of feature extraction.

4 Model Development for Prediction of Rates

In this study, machine learning models have been developed to predict the flow rates of gas, oil, and water in a three-phase flow system. The models have been trained and tested on datasets of historical flow rate measurements from accelerometer readings.

Figure 9 illustrates the schematic diagram of the entire flow rate prediction process. At first as discussed previously in detail, 17 peak values are extracted from each accelerometer data of all the samples. These peak values are used as feature values to predict flow rate. Then an ANN model is trained with the prepared data to predict flow rates.

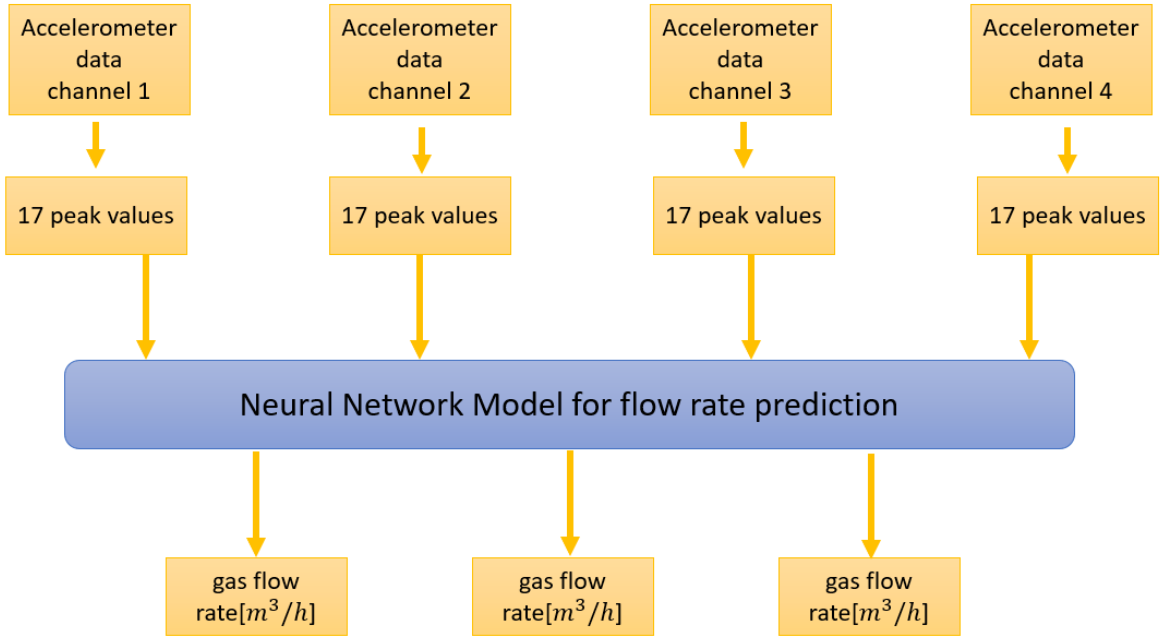

Figure 9: Schematic diagram workflow of flow rate prediction

4.1 Multiphase Flow Rate Prediction

The dataset was preprocessed by normalizing the features and splitting it into training and testing sets. The model was trained using a neural network architecture with multiple hidden layers and optimized using Adam optimization algorithm. The model trains on accelerometer data and predicts flow rates of gas, oil and water simultaneously.

Figure 10 illustrates the architecture of the model used to predict flow rates of multiphase flow. The model utilizes different combinations of dense layer separately for gas, oil and water respectively.

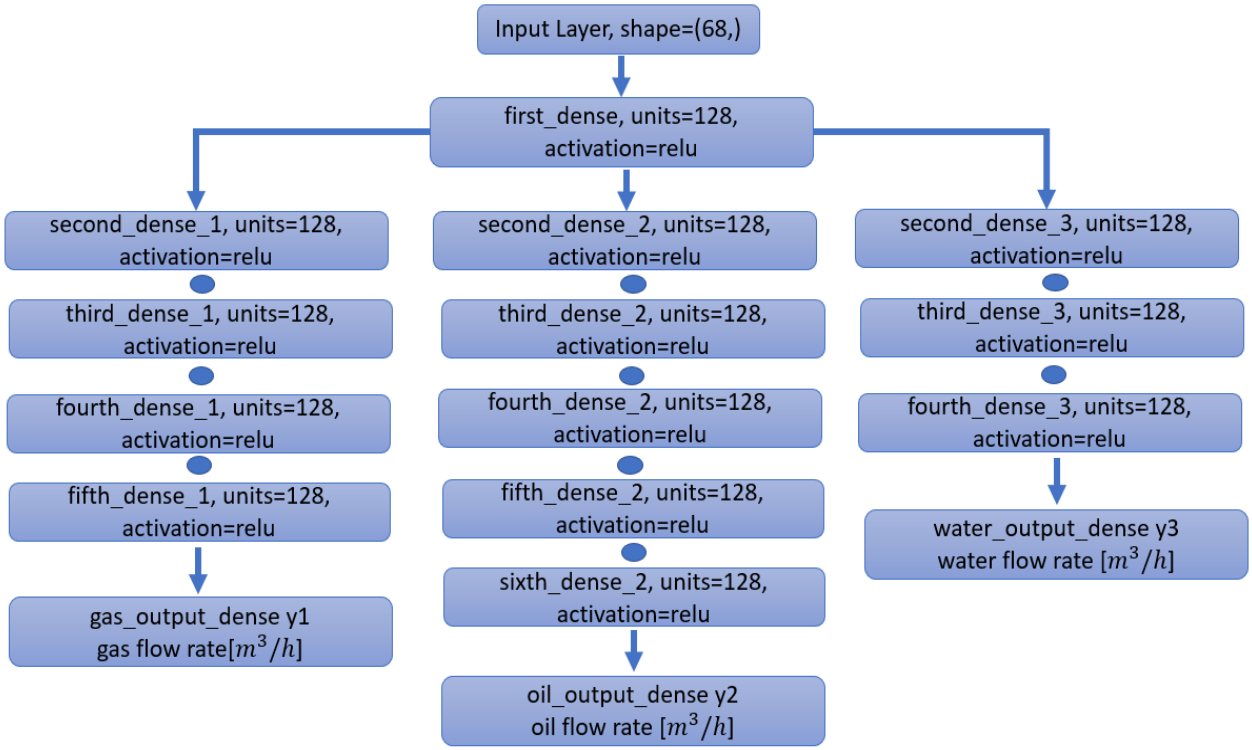

Figure 10: ANN model architecture of multiphase flow prediction model.

The model has in input layer with (68,) shape as there are 68 features in the dataset. First dense layer has 128 neurons with 'ReLU' activation function. From the First dense layer, 3 branches are created for gas, oil and water flow rate prediction. The number of dense layers has been tuned by training and validated numerous times. All the dense layers in the model have the same number of neurons and 'ReLU' activation function except for output layers. 1st branch from the left, which is the gas flow rate prediction branch has dense 4 layers. 2nd branch which is oil branch has 5 dense layers and 3rd branch which is water branch has 3 dense layers before their respective output layers. The output layer for gas is y1, oil is y2 and water is y3. The model has been trained for 8 epochs with a batch size of 2. The code of the multiphase flow prediction neural network model can be found in Appendix H.

4.1.1 Performance evaluation of multiphase prediction model

The performance of the model was evaluated using coefficient of determination (R^2) on the testing set. The results showed that the model was able to predict the flow rates of gas, oil, and water to a satisfactory level by considering the number of available samples, when two or more of the flow rates were non-zero. The accuracy of the predictions decreased significantly when two of the flow rates were zero and only one was non-zero.

Figure 11 displays the training and validation RMSE curves gas, oil and water. The training and validation RMSE curves of all three phases were merged with many attempts by changing the number of layers, neurons, activation function for each phase.

Table 1 represents RMSE values of multiphase flow model predictions on validation dataset. The RMSE values of all three phases are quite satisfactory considering the number of samples used to train the model.

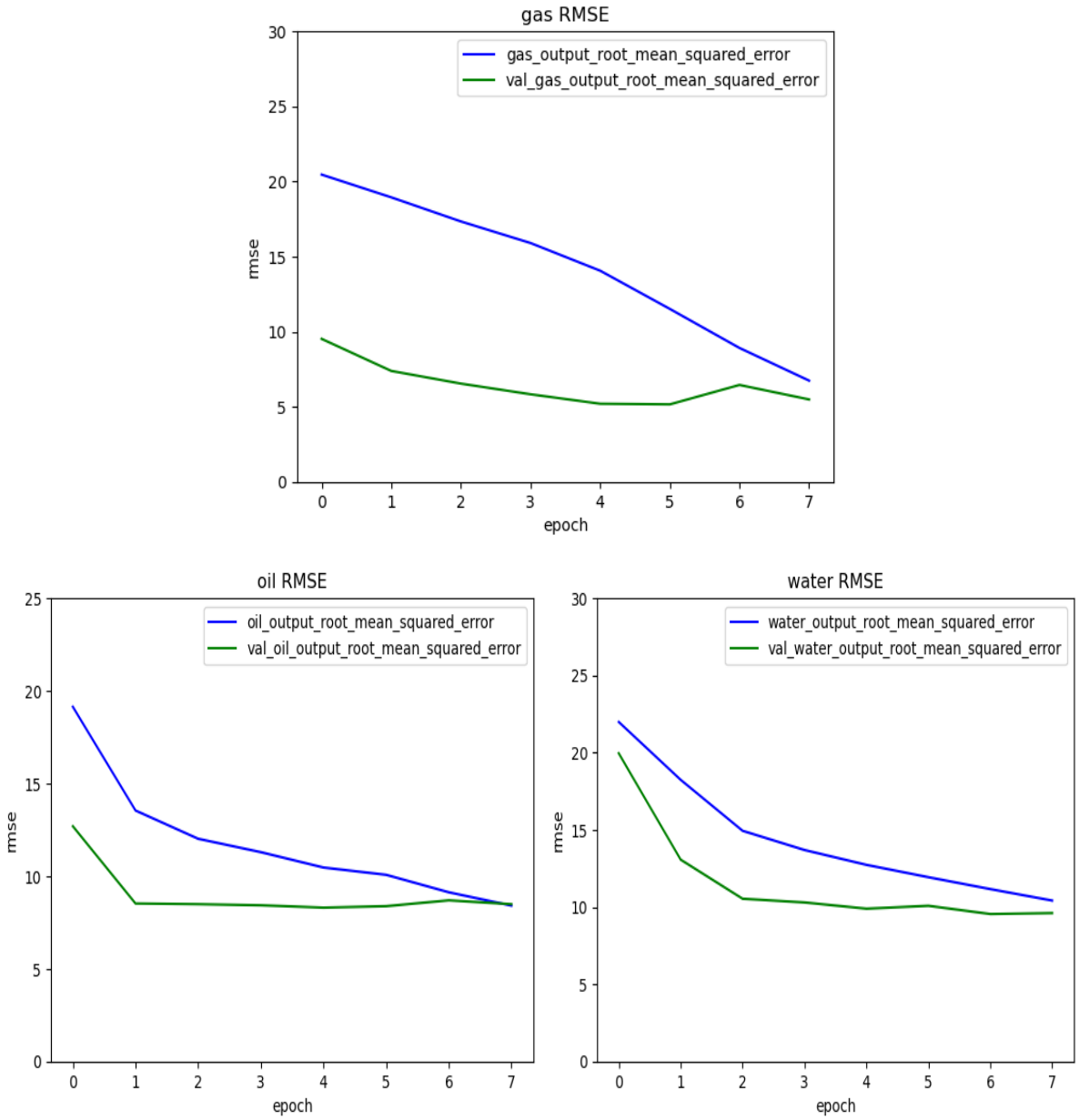

Figure 11: RMSE curves - gas, oil & water – training and validation curves merging at 8th epoch.

Table 1: RMSE values from validation data – Multiphase flow model.

Gas RMSE $[m^3/h]$	5.48
Oil RMSE $[m^3/h]$	8.48
Water RMSE $[m^3/h]$	9.60

It has been observed that the number of single-phase flow samples are far less than the number of three-phase flow and two-phase flow samples. The predictions for three-phase flow appear to be the most accurate, which may be attributed to the fact that the original dataset contains more three-phase flow samples compared to two-phase flow samples and single-phase flow samples.

In this report, oil flow has been indicated as “O”, gas flow as “G”, water flow as “W”, oil and gas flow as “GO”, gas and water flow as “GW”, water and oil flow as “OW”. And three phase flow – gas, oil and water flow has been indicated as “GOW”.

Below are given the number of samples from each accelerometer for the seven classes:

- GOW: 131
- GO: 20
- GW: 9
- OW: 37
- O: 8
- W: 16
- G: 7

It can be observed above that a clear disproportion exists among the classes. Especially the single-phase samples “O”, “W” and “G” have very few samples compared to the rest. As a result, the ANN model is unbale to learn patterns and predict the flow rate for single phase samples. The developed model shows promise for predicting flow rates in three-phase flow systems, but further research is needed to improve its performance in scenarios where two of the flow rates are zero.

4.2 Single-phase flow prediction

To investigate and confirm the reasons of the multiphase flow prediction models’ inability to predict single phase flows, a similar but separate model has been trained exclusively on the single-phase samples. Single-phase sample of gas, oil and water have been separated from the main dataset that contained all the samples.

Figure 12 illustrates the model architecture of the ANN model built to predict flow rates of single-phase flow. The model utilizes different combinations of dense layer separately for gas, oil and water flows respectively.

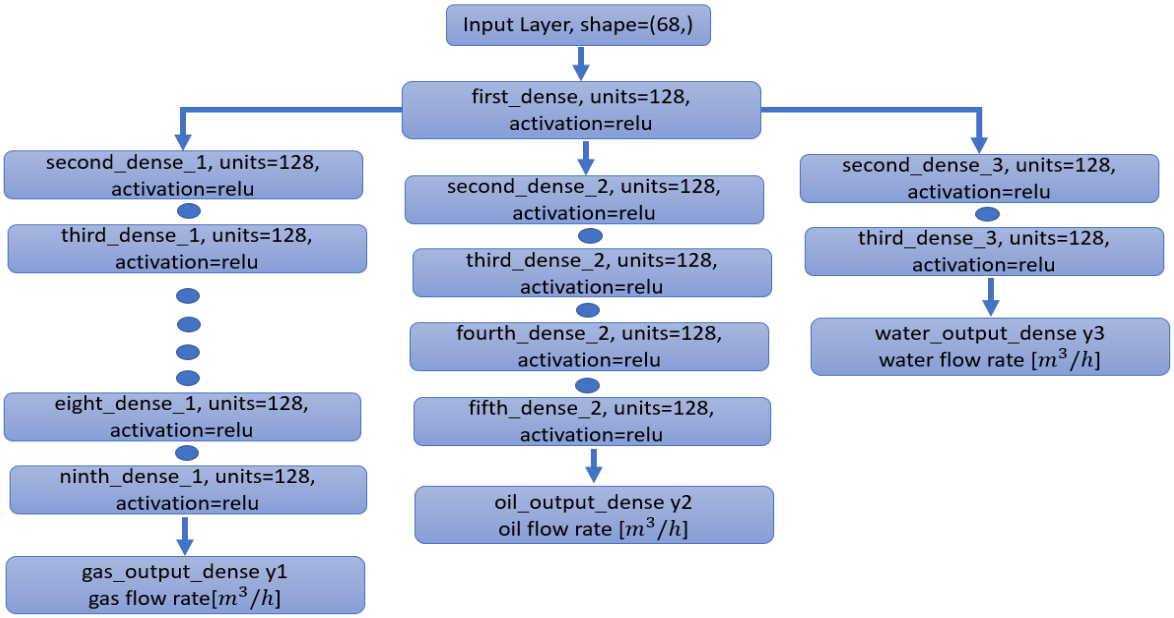

Figure 12: Model architecture of single-phase flow prediction model trained only with single-phase samples.

The model has in input layer with (68,) shape as there are 68 features in the dataset. First dense layer has 128 neurons with 'ReLU' activation function. From the First dense layer, 3 branches are created for gas, oil and water flow rate prediction. The number of dense layers has been tuned by the training and testing numerous times. All the dense layers in the model have the same number of neurons and 'ReLU' activation function except for output layers. 1st branch, which is the gas branch, has 8 dense layers, 2nd branch which is oil branch has 4 dense layers and 3rd branch which is water branch has 2 dense layers before their respective output layers. The output layer for gas is y1, oil is y1 and water is y3. The model has been trained for 5 epochs with a batch size of 2. The code for single-phase flow rate prediction neural network model can be found in Appendix I.

4.2.1 Performance evaluation of single-phase prediction model

In this section, it has been investigated whether the data contains the information required to predict single-phase flow by analyzing single phase prediction model's performance on validation dataset.

Figure 13 illustrates the RMSE plots of only single-phase flow model. After several attempts with different configurations, the training and validation plots could be merged to below extent due lack of enough samples.

Table 2 presents RMSE values of single-phase model predictions on validation data.

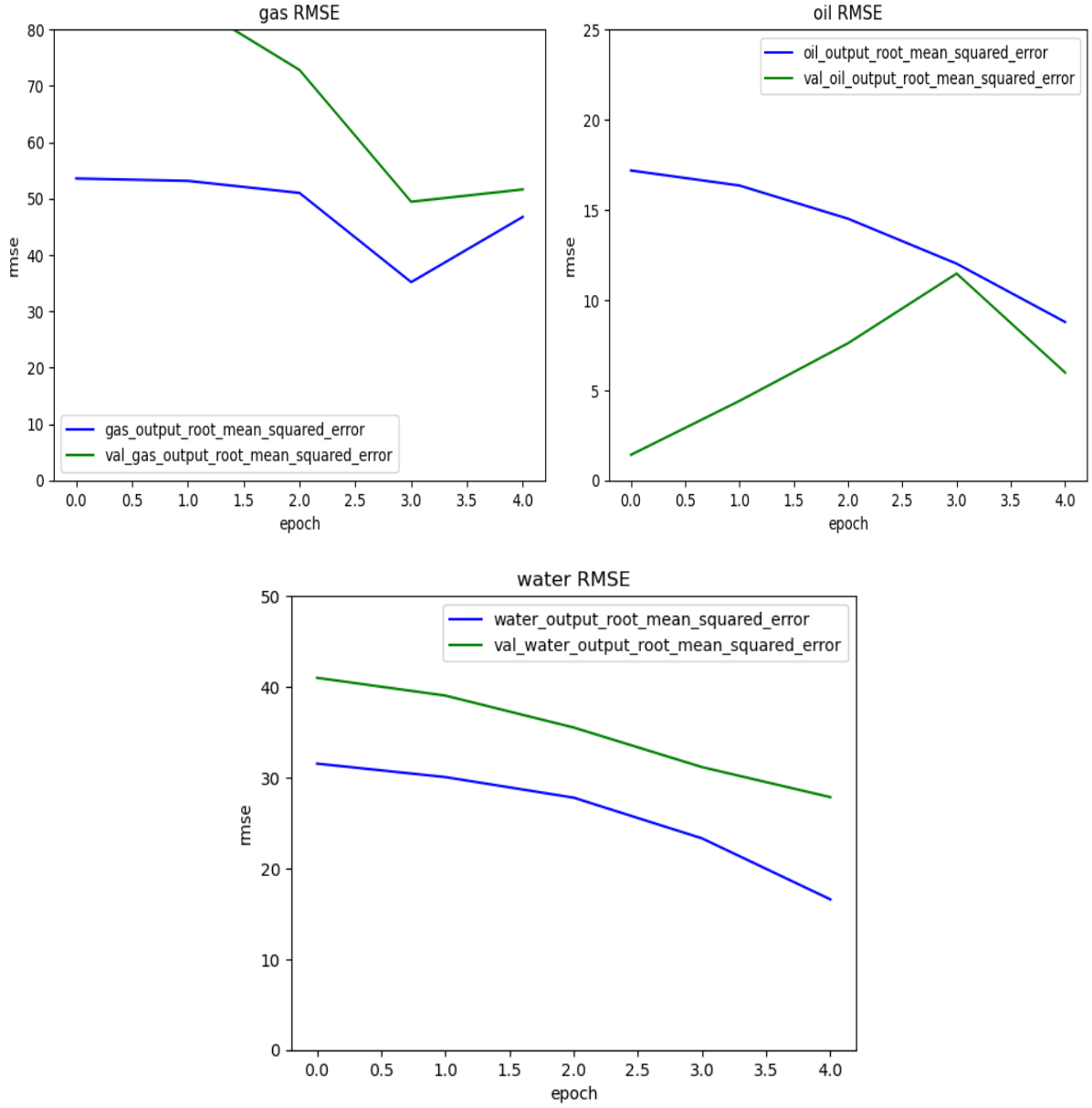

Figure 13: Single phase prediction RMSE - training and validation plots.

Table 2: RMSE values from validation data – Single-phase flow model.

Gas RMSE $[m^3/h]$	51.61
Oil RMSE $[m^3/h]$	5.96
Water RMSE $[m^3/h]$	27.84

4.3 Multiphase flow rate prediction using data from reduced amount (three) of accelerometers

Additional tests have been conducted to examine whether flow rate predictions can be achieved using three accelerometer data instead of four. Similar to previous approaches, the dataset has been preprocessed by normalizing the features and splitting it into training and testing sets. The model has been trained using a neural network architecture with multiple hidden layers and optimized using Adam optimization algorithm. The model trains on accelerometer data and predicts flow rates of gas, oil and water simultaneously. In Figure 14, the model architecture of multiphase flow rate prediction ANN model trained with only three accelerometer data can be observed.

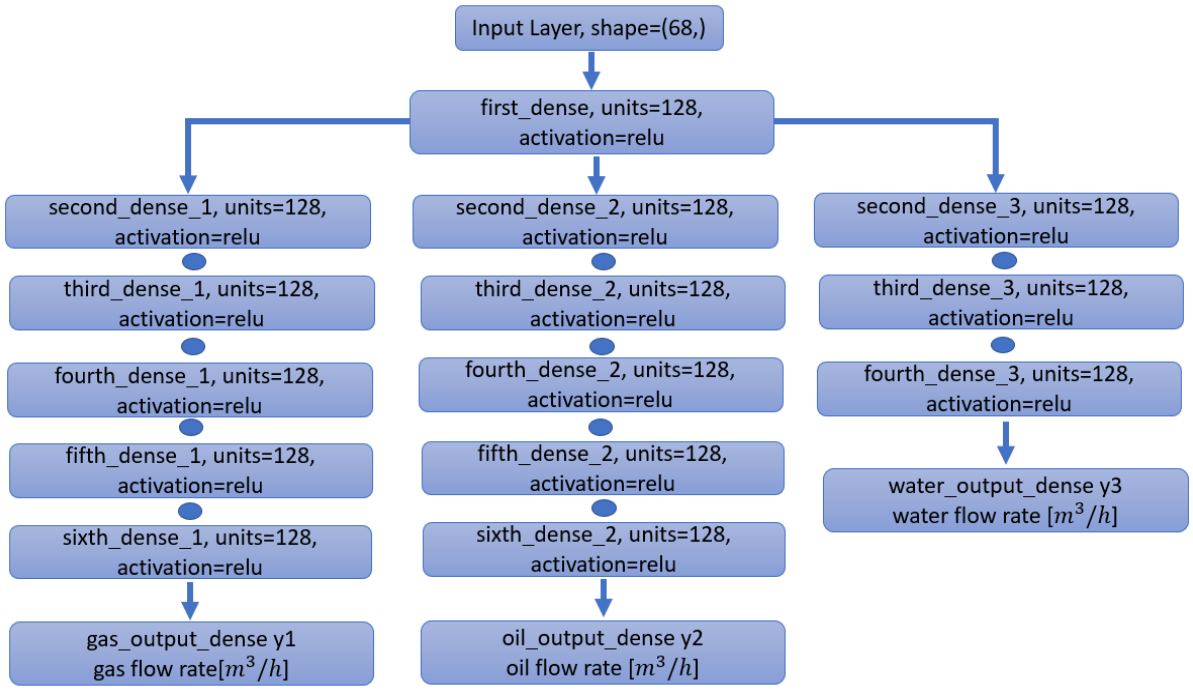

Figure 14: Model architecture of multiphase flow rate prediction model trained with three accelerometer data

The model has in input layer with (68,) shape as there are 68 features in the dataset. First dense layer has 128 neurons with 'ReLU' activation function. From the First dense layer, 3 branches are created for gas, oil and water flow rate prediction. The number of dense layers has been tuned by the training and testing numerous times. All the dense layers in the model have the same number of neurons and 'ReLU' activation function except for output layers. 1st branch, which is the gas branch has 5 dense layers, 2nd branch which is oil branch has 5 dense layers and 3rd branch which is water branch has 3 dense layers before their respective output layers. The output layer for gas is y1, oil is y1 and water is y3. The model has been trained for 7 epochs

with a batch size of 2. Code of multiphase flow prediction neural network model trained with only 3 accelerometer data can be found in Appendix J.

4.3.1 Performance evaluation of multiphase flow rate prediction model trained with three accelerometer data

Figure 15, illustrates the training and validation RMSE curves of multiphase flow prediction model trained on three accelerometer data.

Table 3 represents RMSE values on validation dataset of multiphase flow model trained with three accelerometer data. The RMSE value are higher than they were in case of the model trained with four accelerometer data.

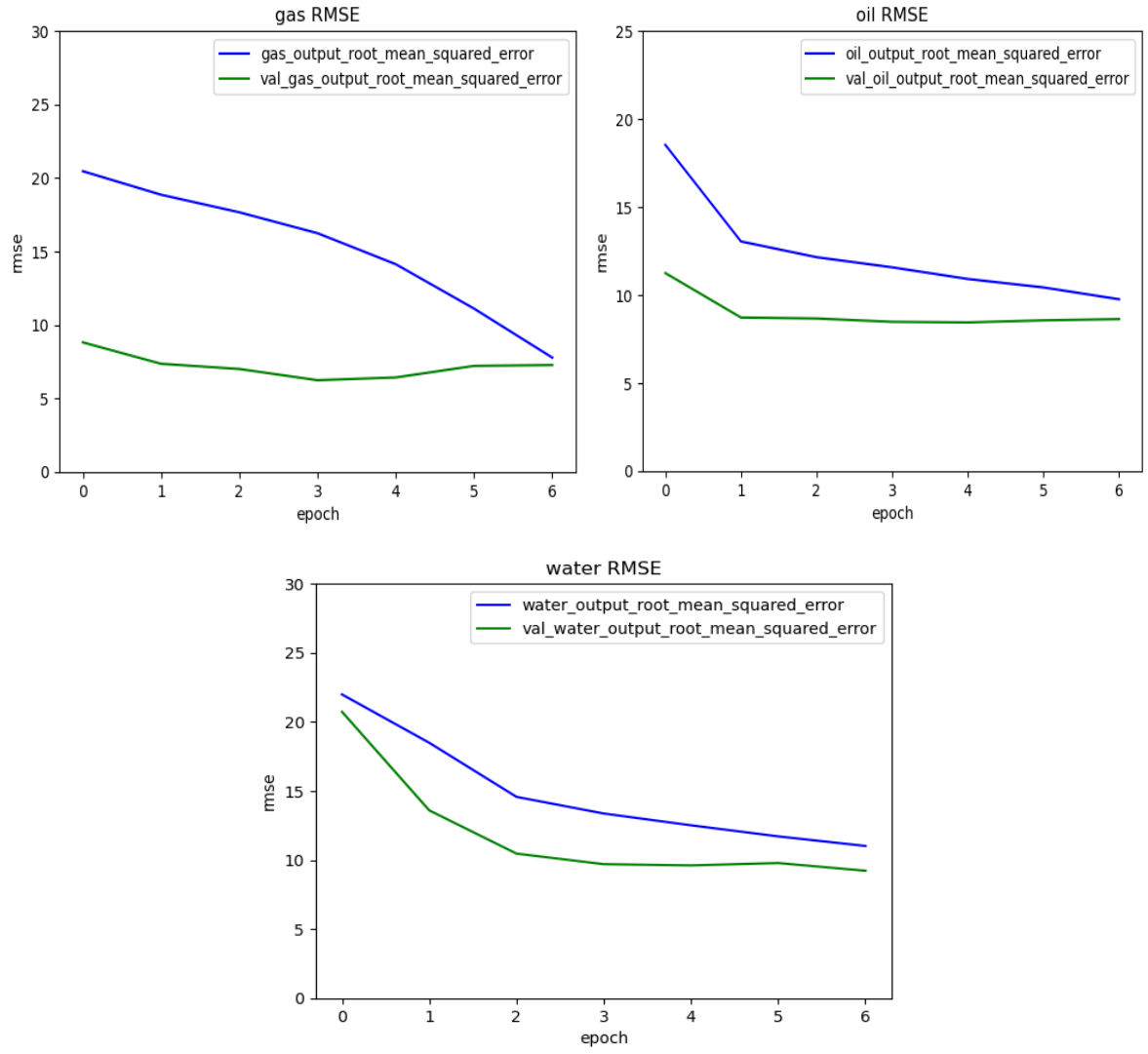

Figure 15: RMSE plots of multiphase flow prediction model trained on 3 accelerometer data.

Table 3: RMSE values from validation data – Multiphase flow model (three accelerometer).

Gas RMSE $[m^3/h]$	7.26
Oil RMSE $[m^3/h]$	8.62
Water RMSE $[m^3/h]$	9.22

4.4 Single-phase flow rate prediction using data from reduced amount (three) of accelerometers

As the model trained with three accelerometer sensor data could not predict single phase samples; the single-phase model approach has been taken again. Figure 16 illustrates the ANN model architecture of single-phase flow rate prediction model trained on three accelerometer data. The model performed best with the same architecture as the single-phase flow rate prediction model trained on four accelerometer data in Figure 16.

The model has in input layer with (68,) shape as there are 68 features in the dataset. First dense layer has 128 neurons with 'ReLU' activation function. From the First dense layer, 3 branches are created for gas, oil and water flow rate prediction. The number of dense layers has been tuned by the training and testing numerous times. All the dense layers in the model have the same number of neurons and 'ReLU' activation function except for output layers. 1st branch, which is the gas branch has 8 dense layers, 2nd branch which is oil branch has 4 dense layers and 3rd branch which is water branch has 2 dense layers before their respective output layers. The output layer for gas is y1, oil is y1 and water is y3. The model has been trained for 5 epochs with a batch size of 2. In Appendix K, the code of single-phase flow prediction neural network model trained with three accelerometer data can be found.

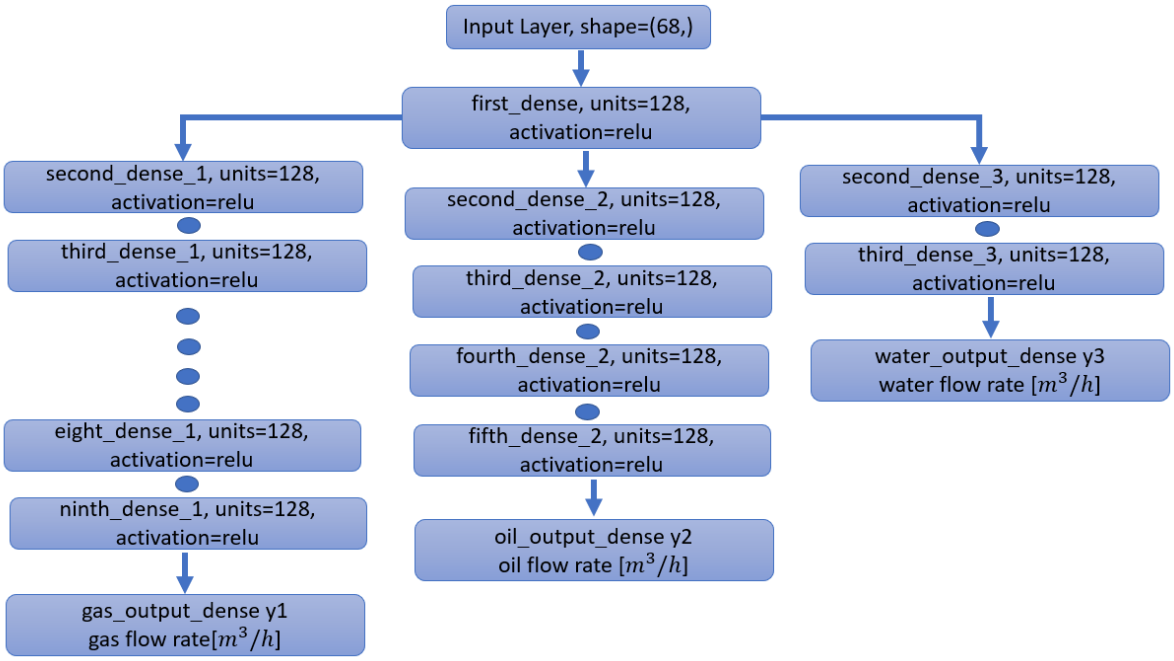

Figure 16: Model architecture of single-phase prediction model trained on three accelerometer data.

4.4.1 Performance evaluation of single-phase flow rate prediction model with three accelerometer data

Figure 17 displays the training and validation RMSE curves of single-phase prediction model trained on three accelerometer data. It appeared to be quite difficult to merge the training and validation curves in this case.

Table 4 shows that RMSE values on validation data of single-phase flow rate prediction model trained on three accelerometer sensor data are worse than the model trained on four accelerometer sensor data.

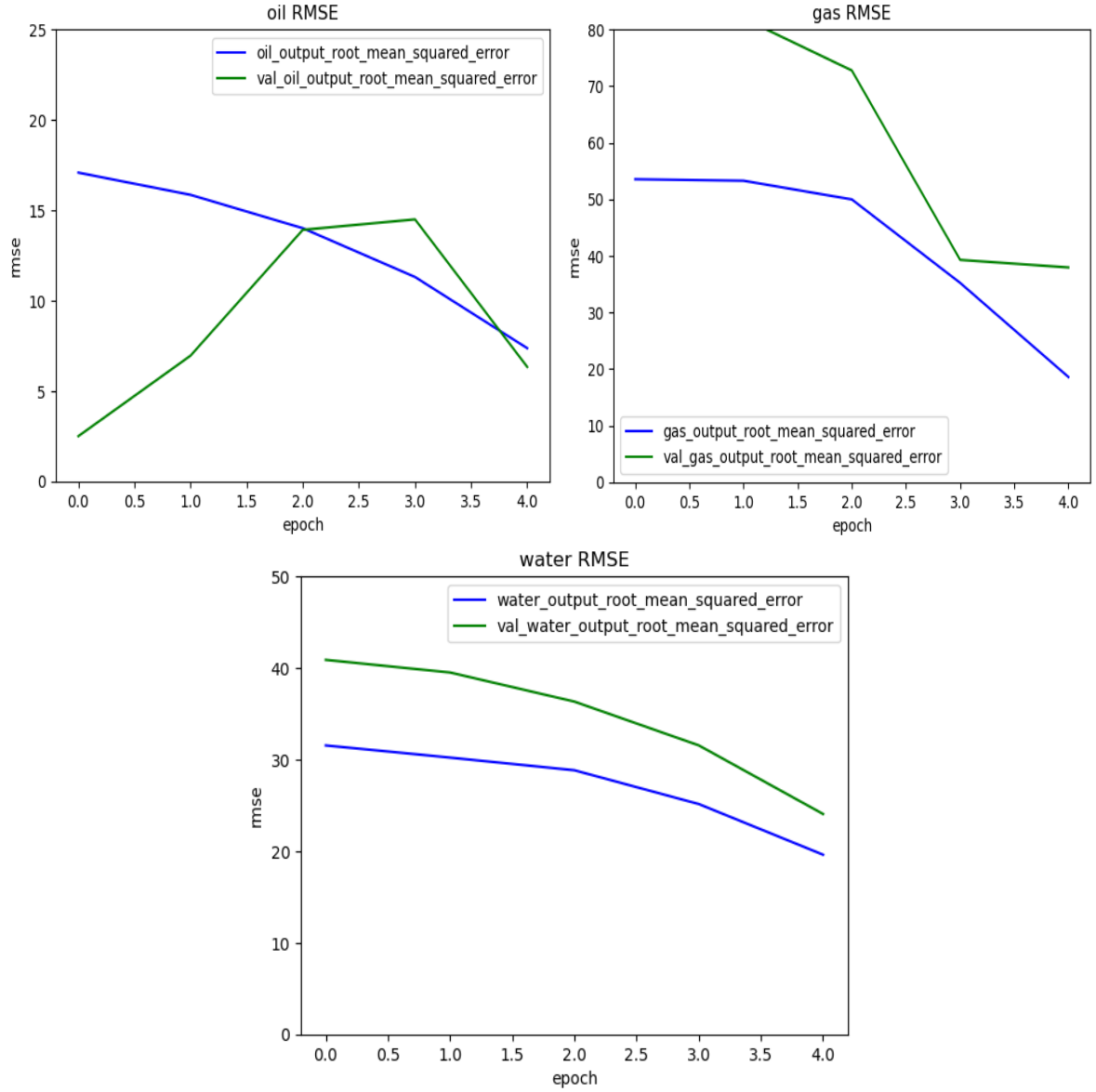

Figure 17: Training and validation RMSE curves – single-phase prediction model trained on three accelerometer data.

Table 4: RMSE values from validation data – Single-phase flow model (three accelerometer).

Gas RMSE $[m^3/h]$	37.93
Oil RMSE $[m^3/h]$	6.32
Water RMSE $[m^3/h]$	24.05

5 Identification of flow types using CNN

The dataset that is being used comprises three distinct phases: gas, oil and water. Each sample represents a unique combination of these phases. The primary objective of this section is to identify the flow type of each individual sample. The fundamental approach involves generating spectrogram images for each sample. And then, a Convolutional Neural Network (CNN) model will be trained and tested using these spectrogram images for the purpose of flow type identification. Figure 18 illustrates the schematic diagram of the proposed flow type classification process.

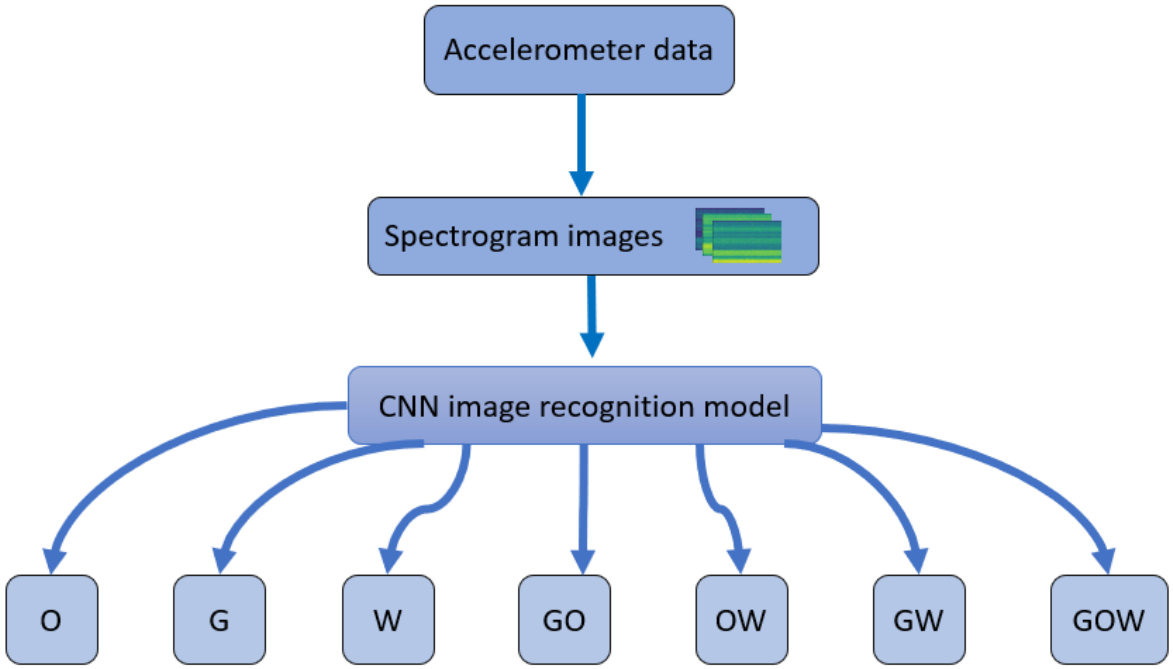

Figure 18: Schematic diagram of flow type classification

5.1 Selection of Significant data points

The dataset used in this thesis comprises seven different flow types, thus creating seven classes for the image recognition model to identify. However, it becomes problematic to utilize all the data available as the imbalanced number of samples in seven classes, potentially leading to bias. For instance, “GOW” has the highest number of sample count at 131, while the “G” has the lowest number of sample count with only 7. To mitigate this issue, the number of samples must be reduced from the classes that have high numbers of samples. To do so, significant samples have been selected from the classes with comparatively higher number of samples. These significant samples are expected to effectively represent their respective classes.

Figure 19, Figure 20 and Figure 21 illustrate the significant samples selection for the GOW, OW, and GO classes respectively.

The selected significant samples have been kept for the task of flow type identification in GOW, OW and GO classes. The other classes already have low number of samples. So, they are kept as they were.

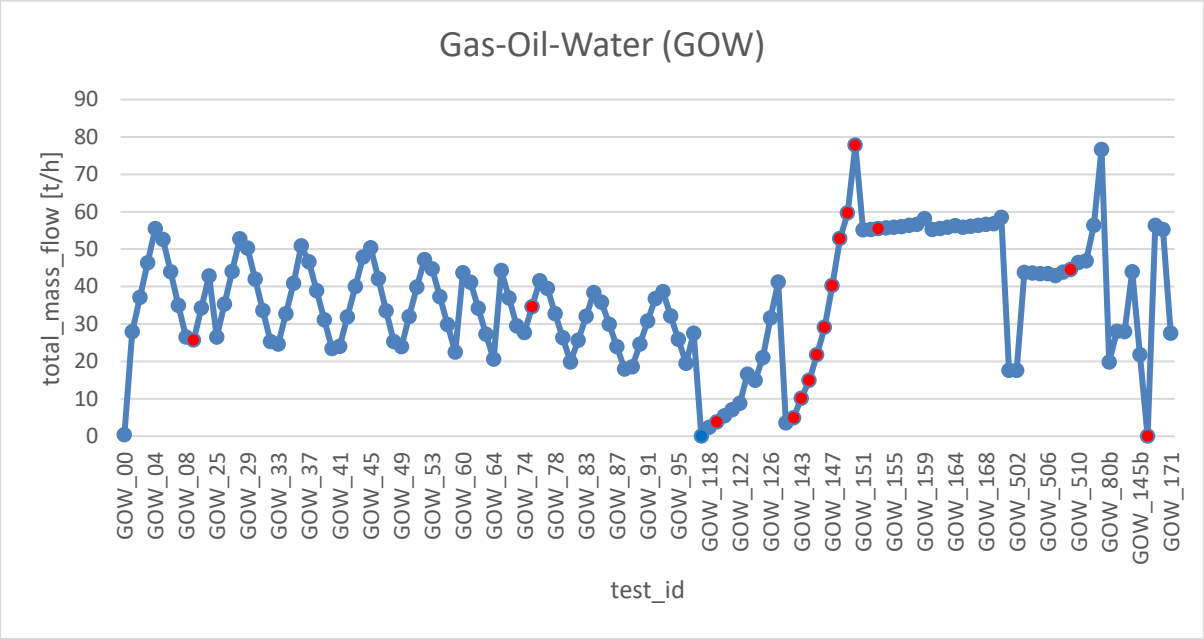

Figure 19: GOW significant samples selection in red.

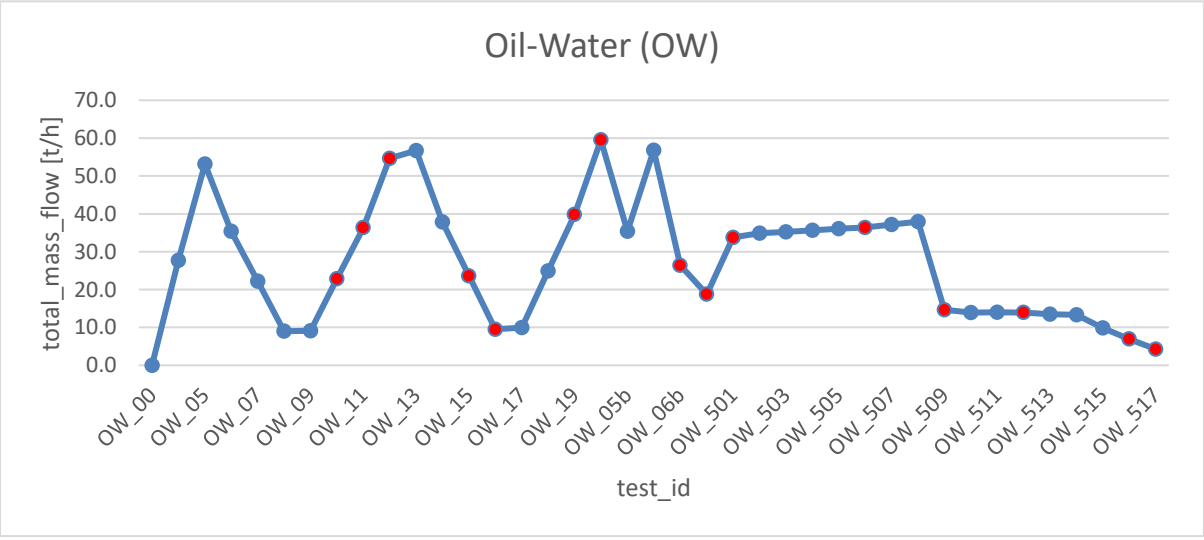

Figure 20: OW significant samples selection in red.

Figure 21

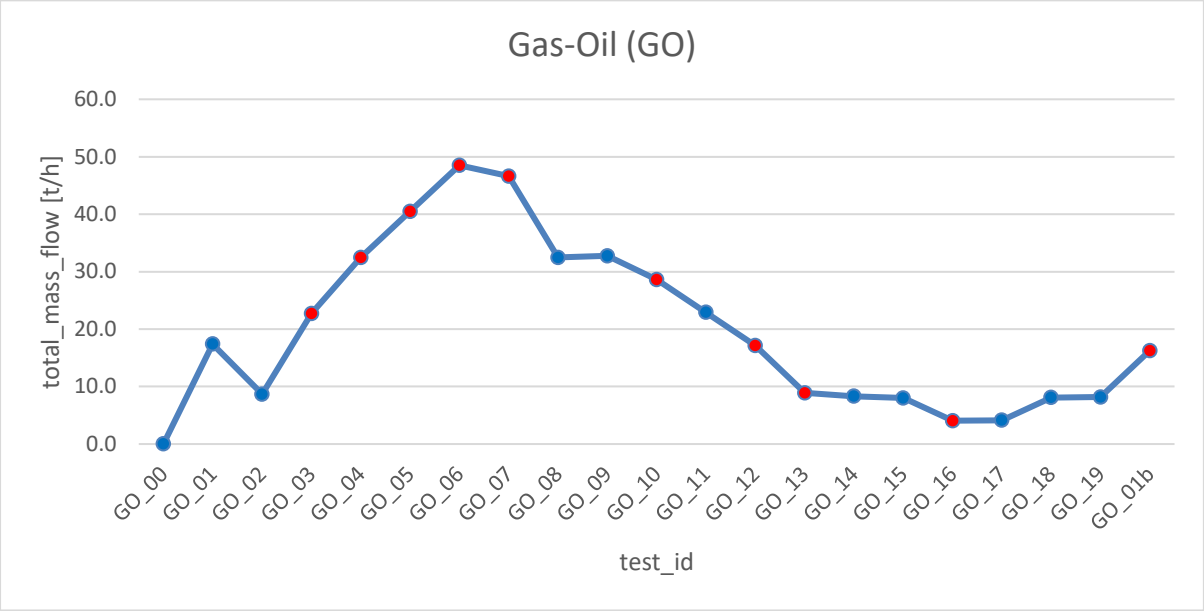

Figure 21: GO significant samples selection in red.

5.2 Spectrogram generation

A python script has been written using ‘signal.spectrogram’ function importing from python ‘Scipy’ library to generate spectrogram images from accelerometer readings. An automated script has been created with the spectrogram generator code inside a loop that goes through each file a specific directory and created spectrogram and then saves the spectrogram in another specified directory.

Spectrograms have been created with several settings before coming to a final format of the spectrogram generation code. Different dynamic ranges have been applied to find one fixed dynamic range that creates clear patterns in all images to be utilized by image recognition model. Figure 22 shows an example spectrogram that has been generated from accelerometer channel 3 data. A CNN model is to be trained with the generated spectrogram images to identify flow types. Spectrograms were generated from all four accelerometer data separately. Among the accelerometer data from four sensors shown in Figure 1 (page 9), accelerometer channel 3 spectrograms have shown the most visible patterns. Thus, spectrograms generated from accelerometer channel 3 data have been used for image recognition purposes.

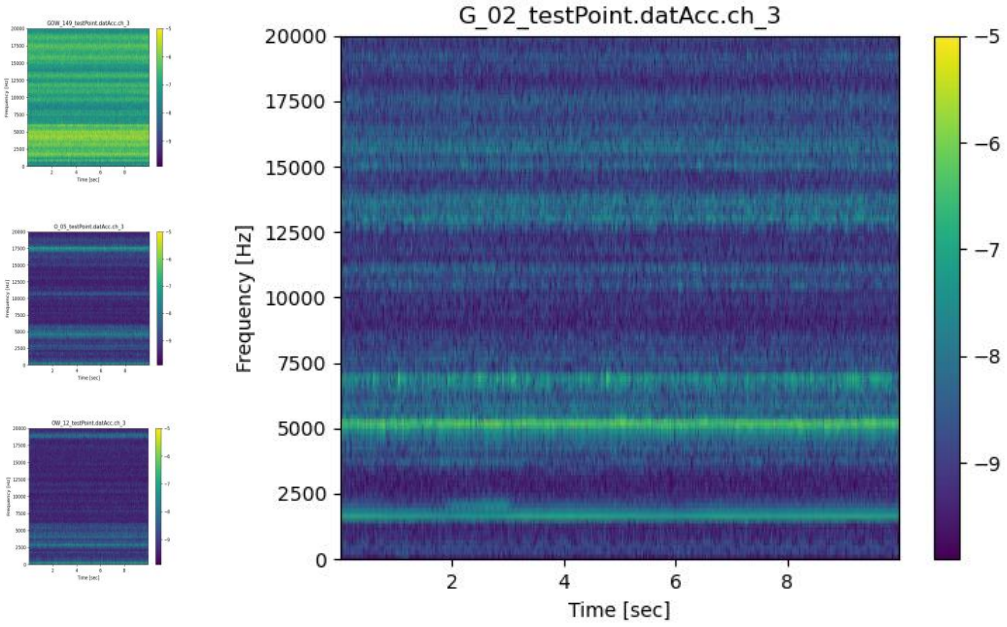

Figure 22: Example of spectrogram generated from accelerometer channel 3 data

5.3 Automated spectrograms generation script

To streamline the image generation process, a python script has been written that takes csv files as input from a specific directory one after another in a loop and generates spectrogram images. The python code for spectrogram generation can be found in Appendix L.

5.4 Data splitting

The spectrogram images of seven classes have been split into a train and test split using python built-in library – 'splitfolders'. Due to having very few samples after creating balance among all classes, after keeping 85% of the samples for training in each class, in some cases 1 or 2 samples are left. For example, class 'G' has only 7 samples. After keeping 5 samples for training, only two samples are left for validation and testing. They need to be kept for testing. But validation data is required as well with which the model will not have been familiar with during training. Considering all these perspectives, the test data set has been used for validation and testing as it will not hamper the training process nor the model performance analysis. The code for splitting the spectrogram dataset is given in Appendix M.

5.5 Image recognition

A CNN image classification technique has been employed to classify the flow types of multiphase flow using the available accelerometer data.

5.5.1 Transfer learning

Due to being low in number of samples, transfer learning is the best solution to construct the Image Recognition model as any transfer learning comes with pre-trained weights on millions of samples, [30].

5.5.2 Tensorflow

While MATLAB provides several in-built transfer learning models, TensorFlow offers an even broader selection. The use of TensorFlow grants access to a multitude of transfer learning models, providing ample opportunities for experimental execution and performance observation on the project's dataset, [31], [32].

5.5.3 Choosing a backbone model

Several backbone models have been tested with the image datasets to select the best one for the image recognition model in the project. A list of performance of the image recognition models with different backbone models with their validation accuracies on the prepared data is given below in Table 5.

Table 5: List of backbone models tried, and accuracies achieved.

Backbone model	Validation accuracy (%)
EfficientNetB0	36.50
ResNet101	33.33
EfficientNetB1	43.33
ResNet101V2	33.33
EfficientNetB2	68.75
EfficientNetB3	50.00
EfficientNetB4	50.50
EfficientNetB5	33.33
EfficientNetB6	36.67

EfficientNetB7	30.45
ResNet152	40.67
ResNet152V2	20.67
EfficentNetB7	36.67
EfficientNetV2B0	50.00
InceptionResNetV2	33.33
EfficientNetV2B2	50.33
EfficientNetV2S	40.35
MobileNet	50.55
EfficientNetV2M	36.67

Keeping the same setup, a backbone model has been replaced to observe the accuracies of the models. EfficientNetB2 has stood out to perform the best with the dataset.

5.5.4 EfficientNetB2

Utilizing Transfer Learning, a pre-trained model – EfficientNetB2 has been used as the backbone of the image recognition model.

EfficientNet is a CNN design and scaling technique that uses a compound coefficient to consistently scale all depth, breadth, and resolution dimensions. In contrast to customary practice, which scales various factors arbitrarily.

In comparison to current CNNs, the EfficientNet models typically attain both higher accuracy and better efficiency. EfficientNetB2 is trained on 14 million ImageNet datasets and predicting 1000 classes. ImageNet dataset constraints 1000 classes. It has been proven to perform more efficiently than many CNNs, [30], [33], [34].

Figure 23, illustrates the model architecture of the image recognition model. The pre-trained weights of the backbone model have been kept and the trainable parameter has been turned off.

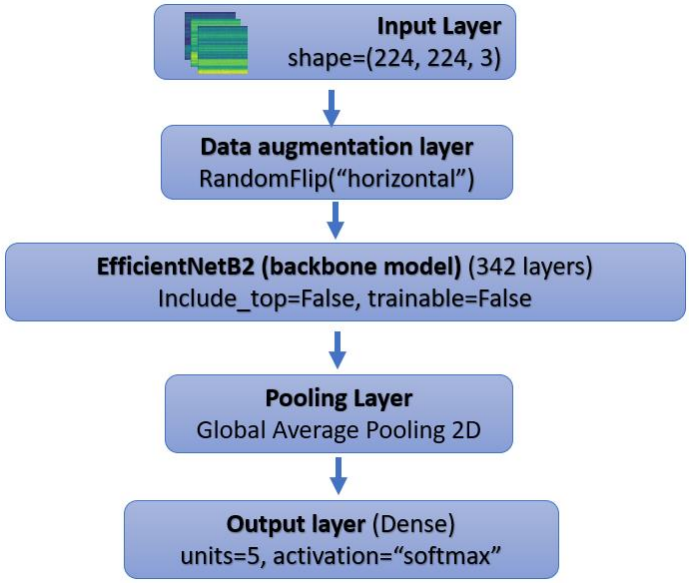

Figure 23: Model architecture of flow type recognition CNN model

Default top layer (1000 neurons for 1000 classes has been turned as 'False' and a custom top layer has been created with 5 neurons for 5 classes. The pre-training of EfficientNetB2 was performed on images with an input shape of (260, 260, 3), [35], [36]. However, for the purpose of this project, modifying the input shape to (224, 224, 3) resulted in improved performance, likely due to the resolution of the input images. An augmentation layer or pre-processing layer has been included in the model to compensate for the low number of samples. Several augmentation techniques have been applied. In the augmentation layer, randomly flipping the images horizontally to artificially increase the number of samples during training has improved the model performance. Global Average Pooling Layer has proven to work best in extracting the most important features from the images. Figure 23 graphically shows brief details of each layer of the image recognition model. The image recognition CNN code for flow type detection of multiphase flow can be found in Appendix N.

6 Physics-based flow rate prediction

To understand the effects of introducing physics into multiphase flow rate prediction models, a customized loss function has been created for an ANN model that is to be trained with accelerometer data that has been prepared in chapter 3 and used in chapter 4 for flow rate prediction.

6.1 Darcy-Weisbach equation

Darcy-Weisbach equation has been used to create the custom loss function. Darcy-Weisbach equation relates head loss due to friction in a pipe to the flow rate and physical properties of the flowing fluid and the pipe. The Darcy-Weisbach equation for head loss can be written as, [37]–[39]:

$$head\ loss, \qquad S = f_d \times \frac{L}{D} \times \frac{v^2}{2g} \tag{i}$$

where,

fd is the Darcy friction factor, which depends on the roughness of the pipe and the Reynolds number.

L is the length of the pipe or duct.

D is the diameter of the pipe or duct.

v is the velocity of the fluid.

g is the acceleration due to gravity.

6.2 Loss function design

In equation (i), Darcy friction factor (f_d) is a complicated parameter, but it is unitless. To avoid introducing complications into the model, friction factor (f_d) is ignored in the loss function in equation (ii). To modify this equation for use as a loss function, velocity (v) of fluid has been designated to present the difference between true flow rate value (y_true) and predicted flow rate value (y_pred). The length (L) has been designated as the distance between accelerometers which is 4 meters. D is the diameter of the rig which is 3 inches or 0.0762 meters. g is the gravitational acceleration, 9.8 m/s^2 . Thus the final loss function is:

$$head\ loss, \qquad S = \frac{L}{D} \times \frac{v^2}{2g} \tag{ii}$$

where,

L = 4 m

D = 0.0762 m

v = (y_true – y_pred) m/s

g = 9.8 m/s^2

6.3 Physics-based Multiphase flow prediction

Figure 24 illustrates the model architecture of physics introduced multiphase flow prediction ANN model. The model has been tuned after introducing a custom loss function utilizing Darcy-Weisbach equation.

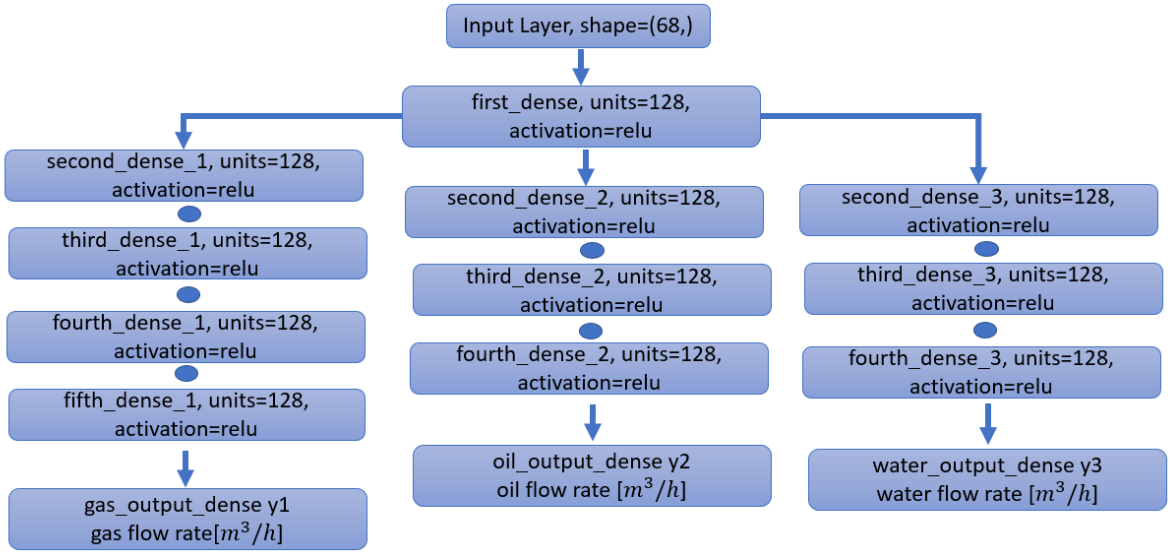

Figure 24: Model architecture of physics-based multiphase flow rate prediction ANN model.

The model has been trained for 18 epochs with batch size of 2. The code for the custom loss with function with ANN model for physics-based multiphase flow rate prediction is given in Appendix O.

6.3.1 Performance evaluation of physics-based flow prediction

Figure 25 displays the RMSE curves of physics-based multiphase flow model. The training and validation curves have merged smoothly in this case.

Table 6 shows the RMSE values of physics-based multiphase flow model predictions. The RMSE values that have been achieved due to the custom loss function are quite similar to usual ANN multiphase flow rate prediction model RMSE values in chapter 4.1.

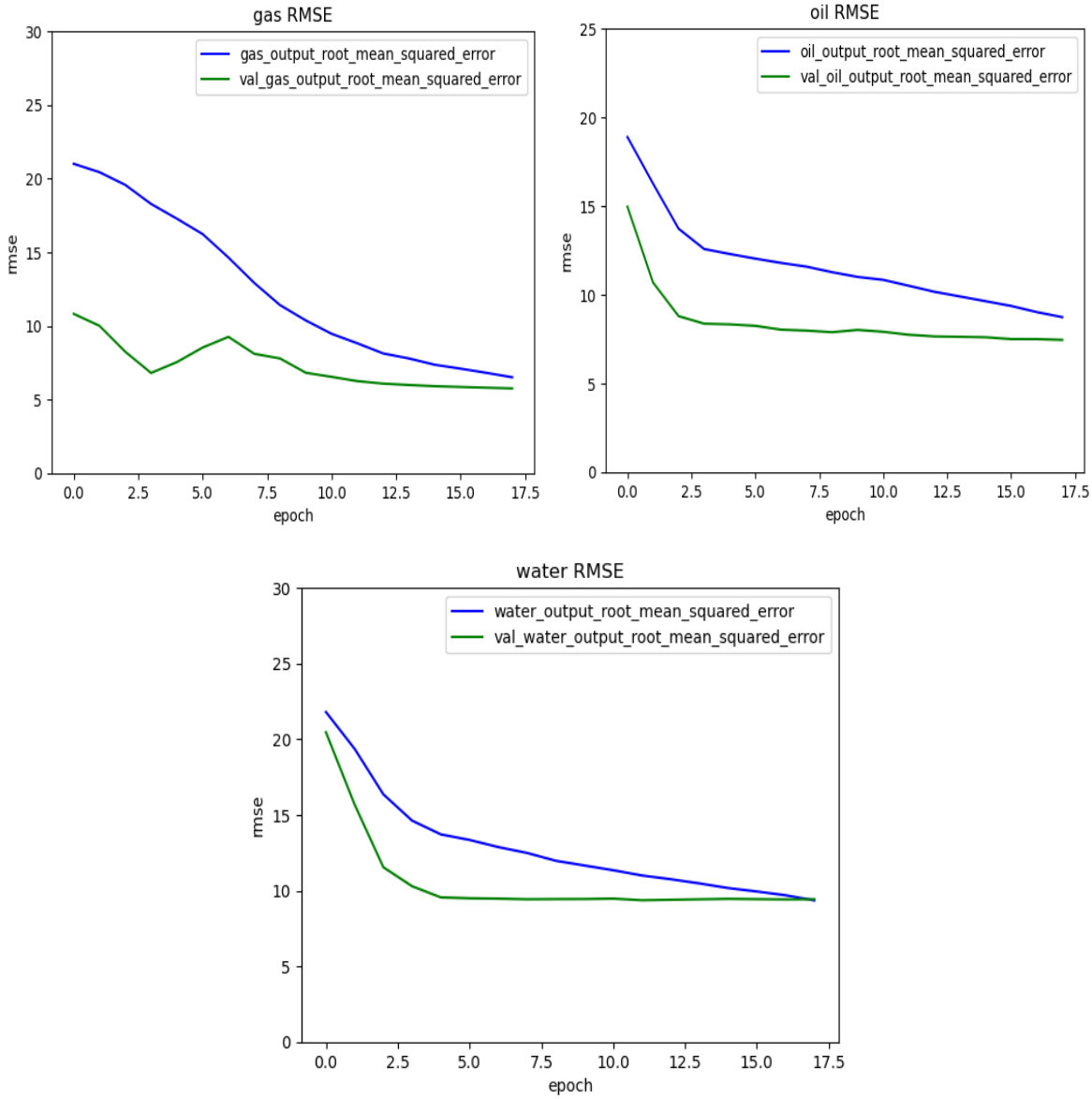

Figure 25: RSME curves of physics-based multiphase flow model.

Table 6: RMSE values from validation data – Physics-based multiphase flow model

Gas RMSE $[m^3/h]$	16.87
Oil RMSE $[m^3/h]$	13.76
Water RMSE $[m^3/h]$	13.85

7 Results

This chapter presents the prediction and classification outcomes of the models discussed so far. Prediction values with true values are analyzed in regression models and probable reasons for obtaining the results are discussed. Similarly true labels with predicted labels are analyzed for classification model and the obtained results are discussed.

7.1 Results of multiphase flow prediction model

Table 7 represents RMSE values of multiphase flow model predictions on test dataset. The RMSE values of all three phases are quite satisfactory considering the number of samples used to train the model.

Table 7: RMSE values from test data predictions – Multiphase flow model.

Gas RMSE $[m^3/h]$	16.00
Oil RMSE $[m^3/h]$	12.22
Water RMSE $[m^3/h]$	13.95

In Figure 26, the regression plots for gas, oil and water are presented. The limitations of the model's predictions can be observed directly in these plots. In the oil and water regression plots, some true values are 0, but the model is unable to predict them with satisfactory accuracy. These points have created sort of horizontal lines in the oil and water regression plots. Other predictions seem to exist around the regression line.

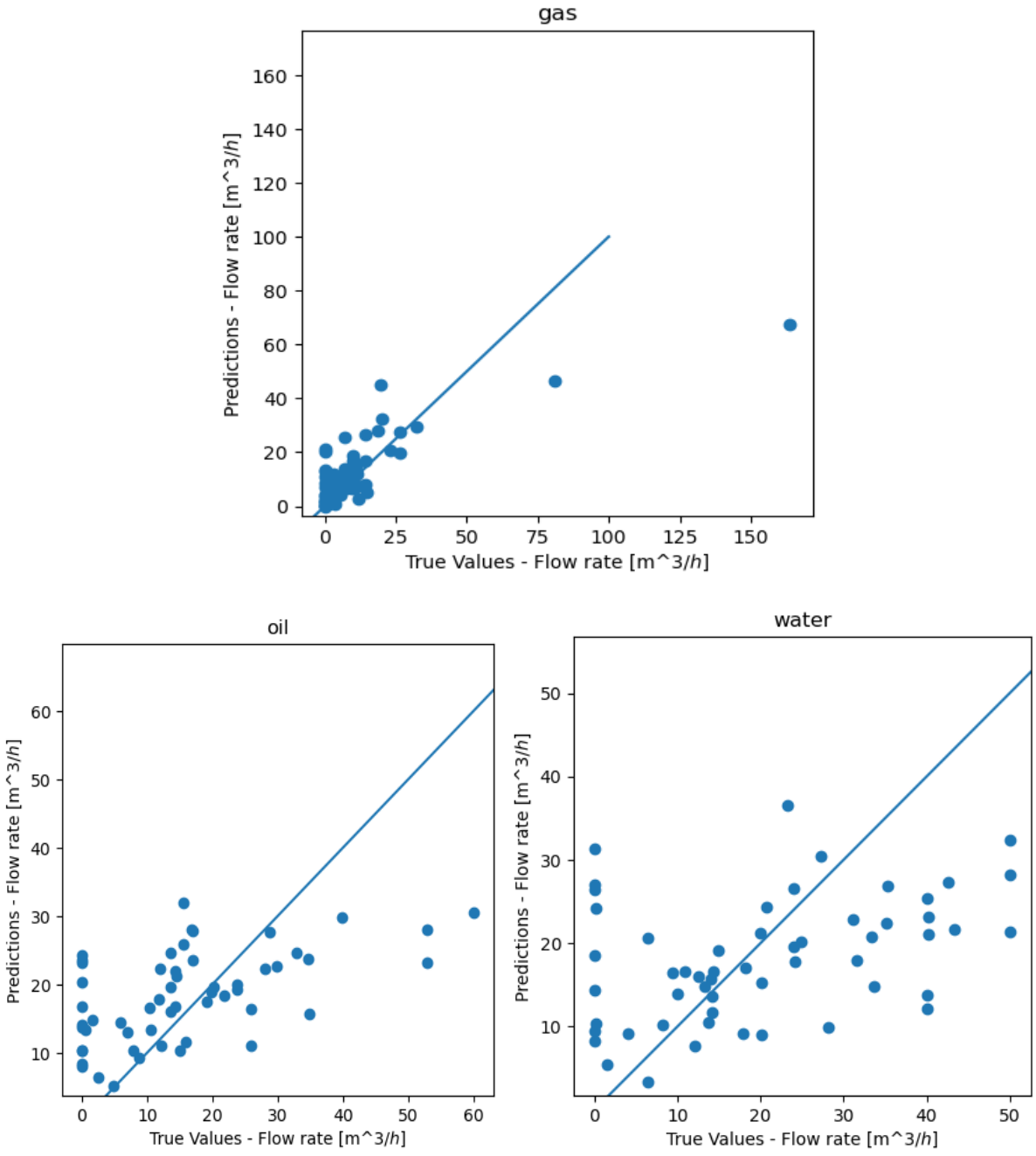

Figure 26: Regression plots of gas, oil and water predictions.

Figure 27, Figure 28 and Figure 29 illustrate the predictions of gas, oil and water flow rate respectively on test dataset from multiphase flow rate prediction ANN model. The model predicts three-phase, two-phase and single-phase flows.

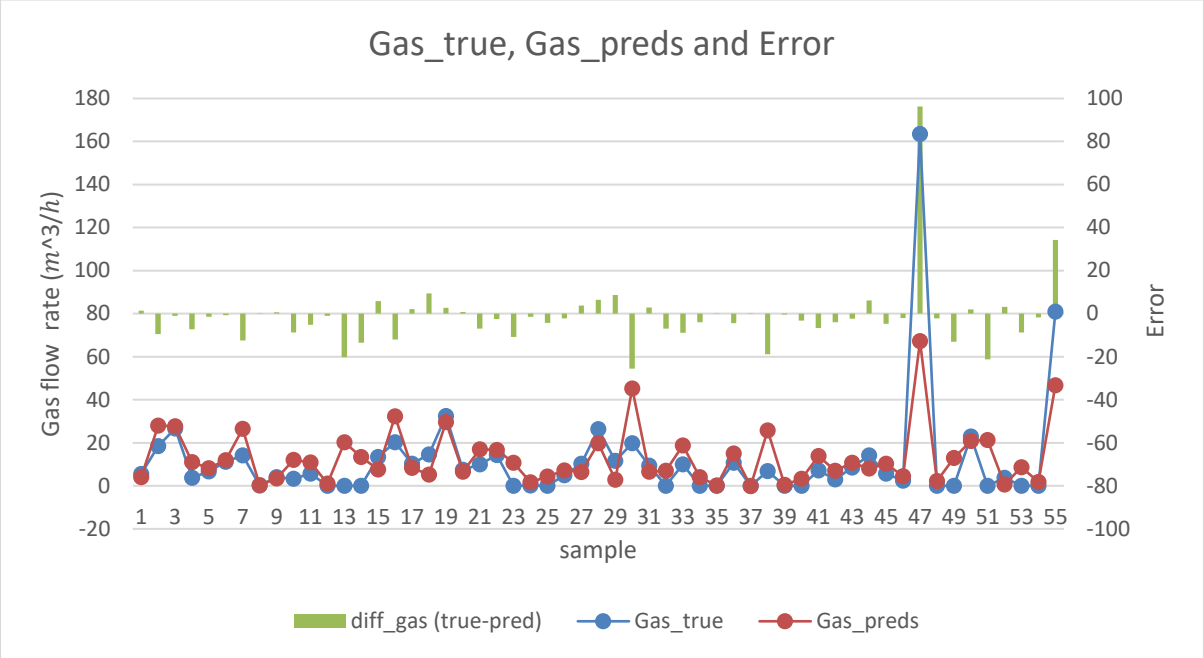

Figure 27: Gas true values and predicted values comparison – Multiphase flow test data.

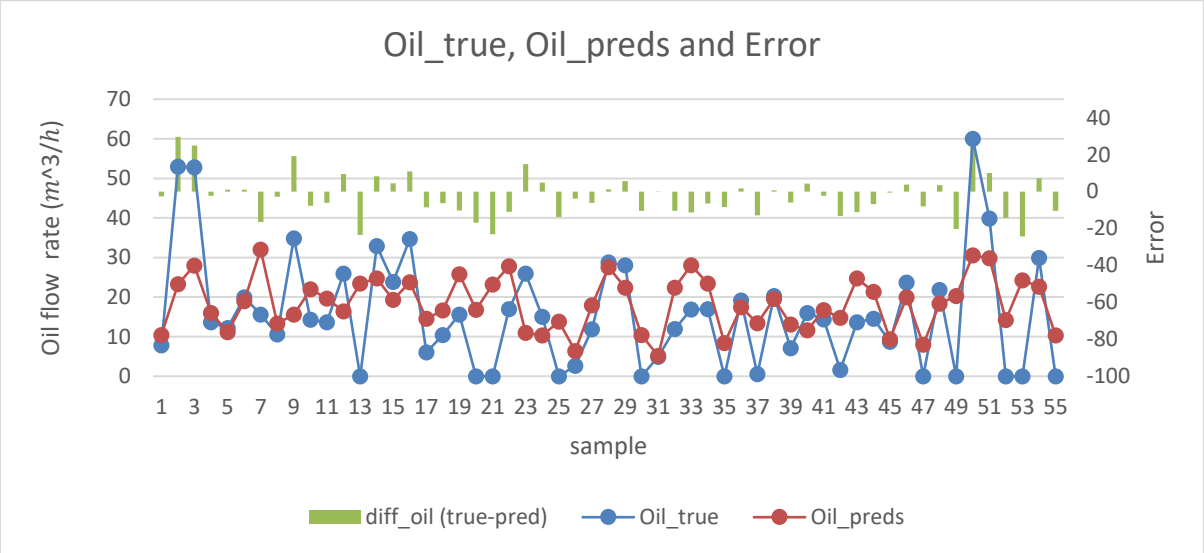

Figure 28: Oil true value and predicted values comparison - Multiphase flow test data.

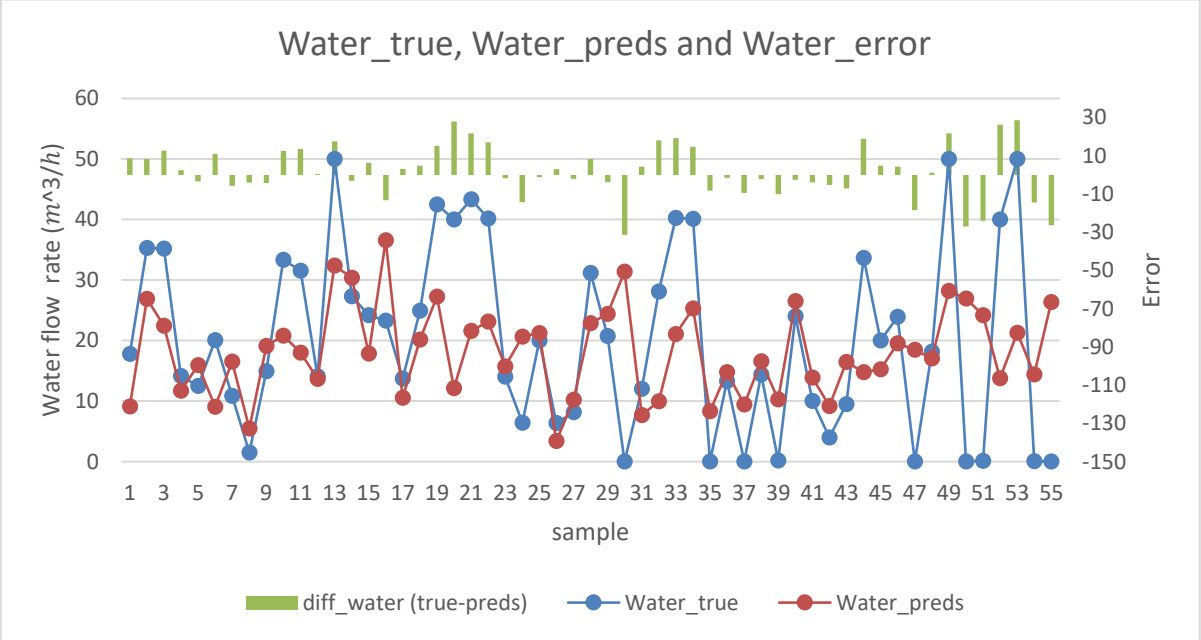

Figure 29: Water true values and predicted values comparison - Multiphase flow test data.

Looking at the predictions in Figure 27, Figure 28 and Figure 29, it is observed that among gas, oil and water, if three or two of the flows are non-zero, the predictions are quite acceptable as experimental values, but when two of the flows are zero and one is phase non-zero in a sample then the predictions deviate far from true values e.g. sample 52. Several three phase models have been created and trained but none of them have been able to predict the single-phase flows with satisfactory performance. Sample 47 has very high flow rate but there were not many samples with high flow rates as such for the model to train on; thus, the model fails to predict flow rates for such samples. In Appendix B, a table has been attached that contains the gas, oil and water true values and predicted values with errors from multiphase flow model.

7.2 Results single phase flow prediction model

Table 2 presents RMSE values of single-phase model predictions on test data.

The regression plots of gas, oil and water predictions in Figure 30, shows that model trained with only single-phase flow samples can find patterns in the data and make some predictions that are better than the ones observed in three-phase flow model for single-phase flow samples. Due to very few numbers of single-phase flow samples, further improvement could not be achieved.

Figure 31, Figure 32 and Figure 33 illustrate the predictions of gas, oil and water respectively from single-phase flow rate prediction ANN model on test dataset. The model makes prediction on only single-phase flows. The single-phase predictions values are far better than the last model that had more than one non-zero phase. Though the predictions are not very accurate

but still this proves that the accelerometer data does contain the information that can help ML models learn patterns to predict single phase flows as well.

Table 8: RMSE values from test data predictions – Single-phase flow model.

Gas RMSE $[m^3/h]$	33.08
Oil RMSE $[m^3/h]$	6.59
Water RMSE $[m^3/h]$	16.17

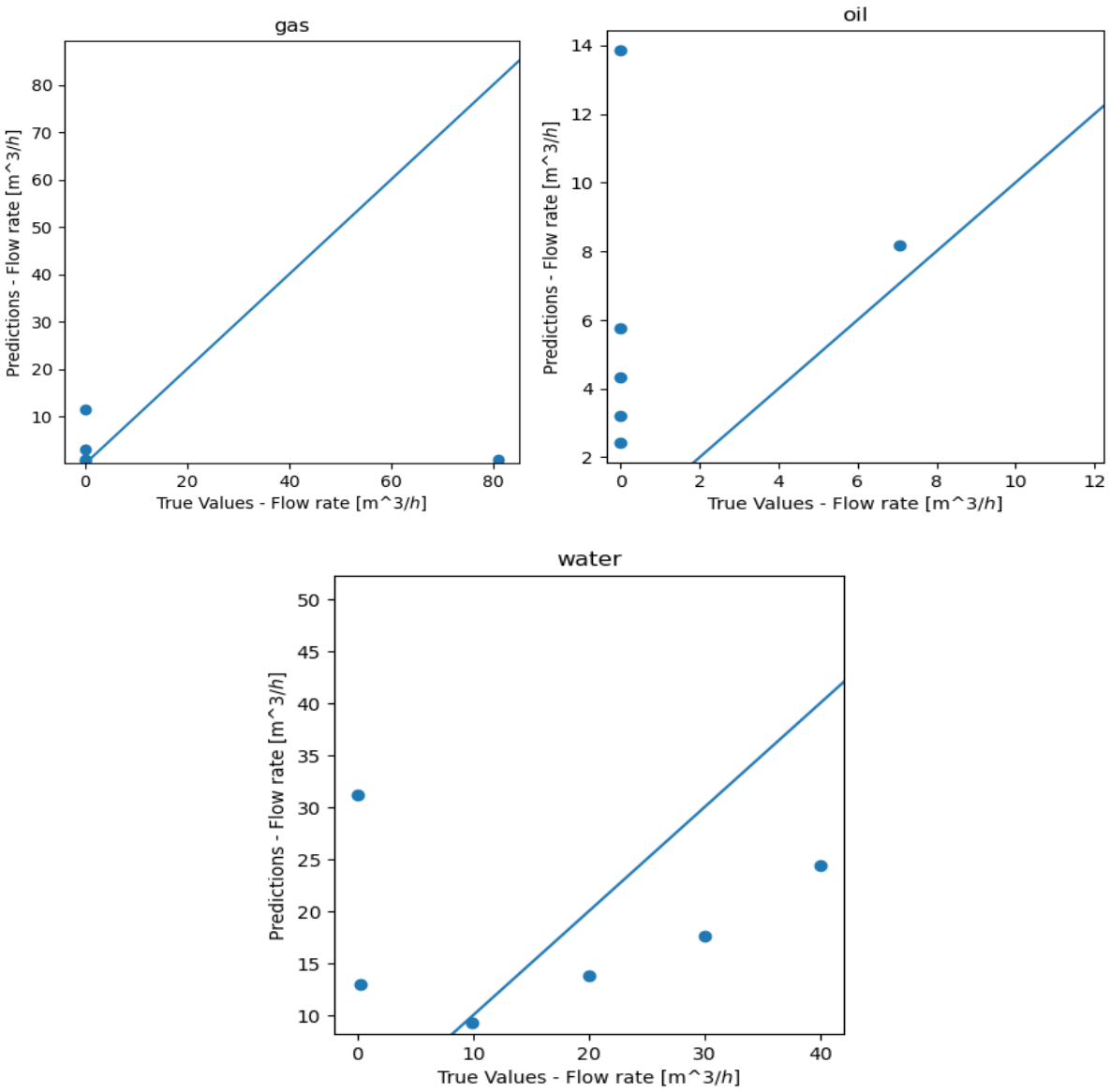

Figure 30: Regression plots of single-phase flow rate predictions.

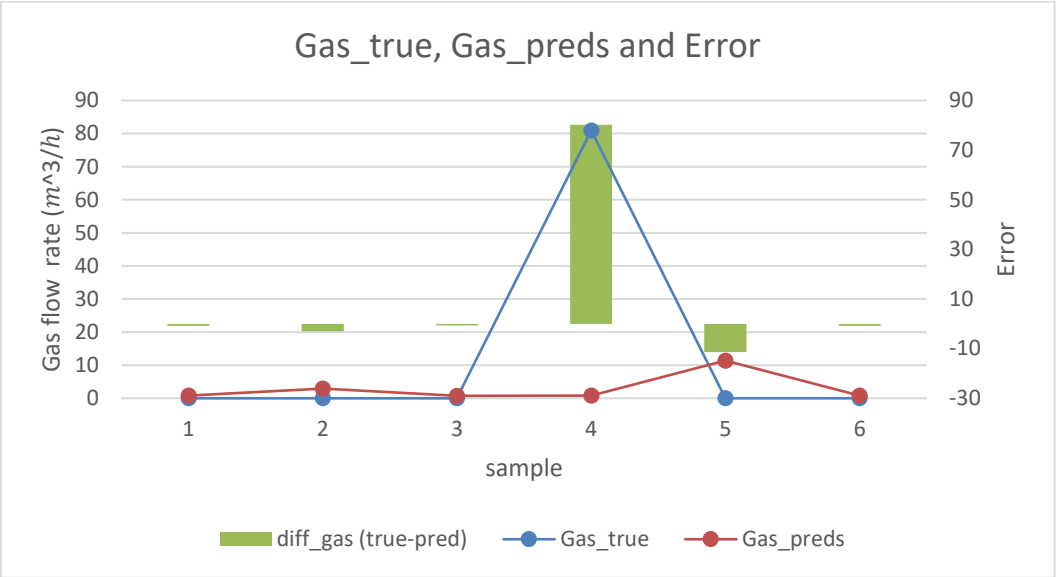

Figure 31: Gas true values and predicted values comparison – Single-phase flow test data.

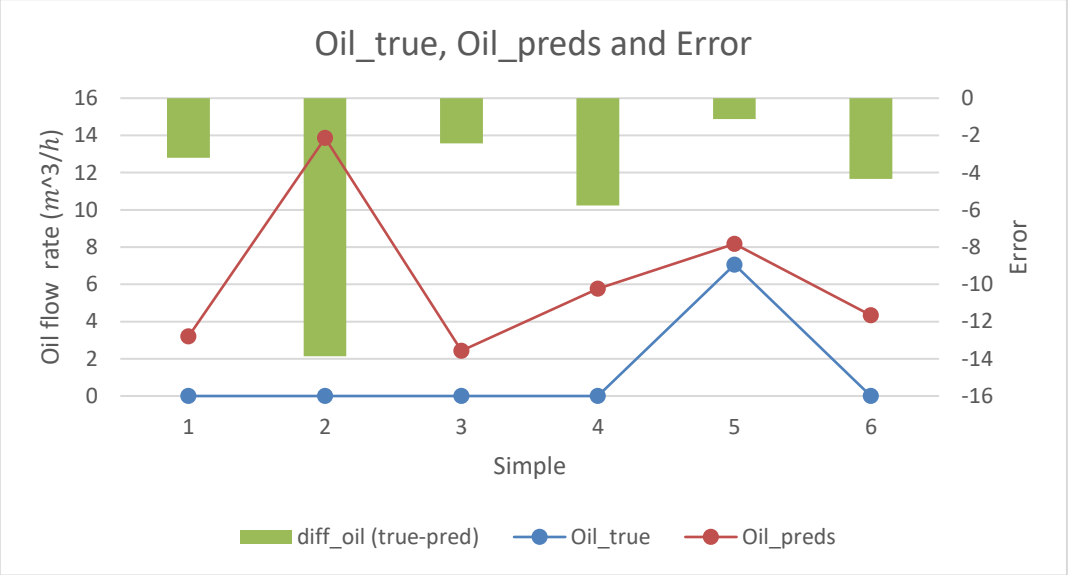

Figure 32: Oil true value and predicted values comparison - Single-phase flow test data.

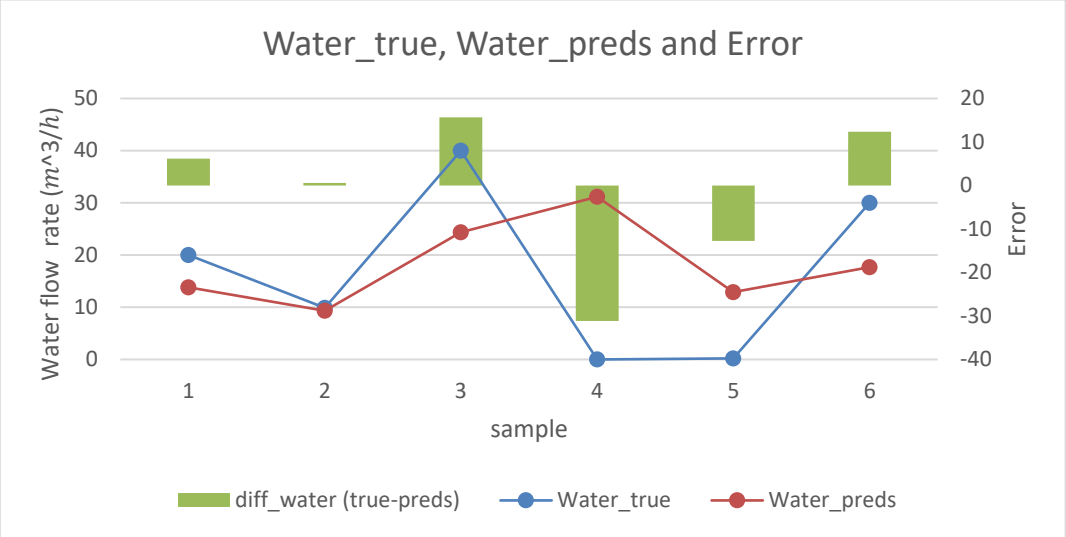

Figure 33: Water true value and predicted values comparison - Single-phase flow test data.

The reason why the predictions are not accurate when two of the flows are zero and one is non-zero could be due to several factors. One possibility is that the model was not trained on enough data points where two of the flows are zero, which can result in inaccurate predictions when these conditions occur in the test data. Another possibility is that the model architecture may not be able to capture the complex relationships between the variables when two of the flows are zero. But the above single phase flow rate experiment provides enough evidence that if there are enough samples to learn from; the models can predict single phase flow rates as well. Sample 4 in Figure 31 and sample 2 in Figure 32 have very high true value. Due to having low number of samples with such high flow, models are predicting poorly in such cases. Appendix C contains the true values and predictions from single-phase model.

7.3 Results of predicting multiphase flow rate using three accelerometers

Table 9, represents RMSE values of multiphase flow model predictions. On test dataset. The RMSE values are higher than they were in case of the model trained with four accelerometer data.

In Figure 34, the regression plots of gas, oil and water flow rate predictions from multiphase flow model trained by using three accelerometer data can be observed. The same fault as the model trained on four accelerometer data can be observed right in the regression plots. In oil and water regression plot, some true values are 0, but the model is unable to predict them with acceptable numbers. These points have created sort of horizontal lines in the oil and water regression plots. Other predictions seem to exist around the regression line.

Figure 35, Figure 36 and Figure 37 shows that the three phase and two-phase sample predictions seem to have improved compared to the model trained with four accelerometer data. But the single-phase sample predictions have worsened.

Appendix D contains the true values, predictions and errors from multiphase flow model trained with three accelerometer data.

Table 9: RMSE values from test data – Multiphase flow model (three accelerometers).

Gas RMSE $[m^3/h]$	18.18
Oil RMSE $[m^3/h]$	13.56
Water RMSE $[m^3/h]$	13.48

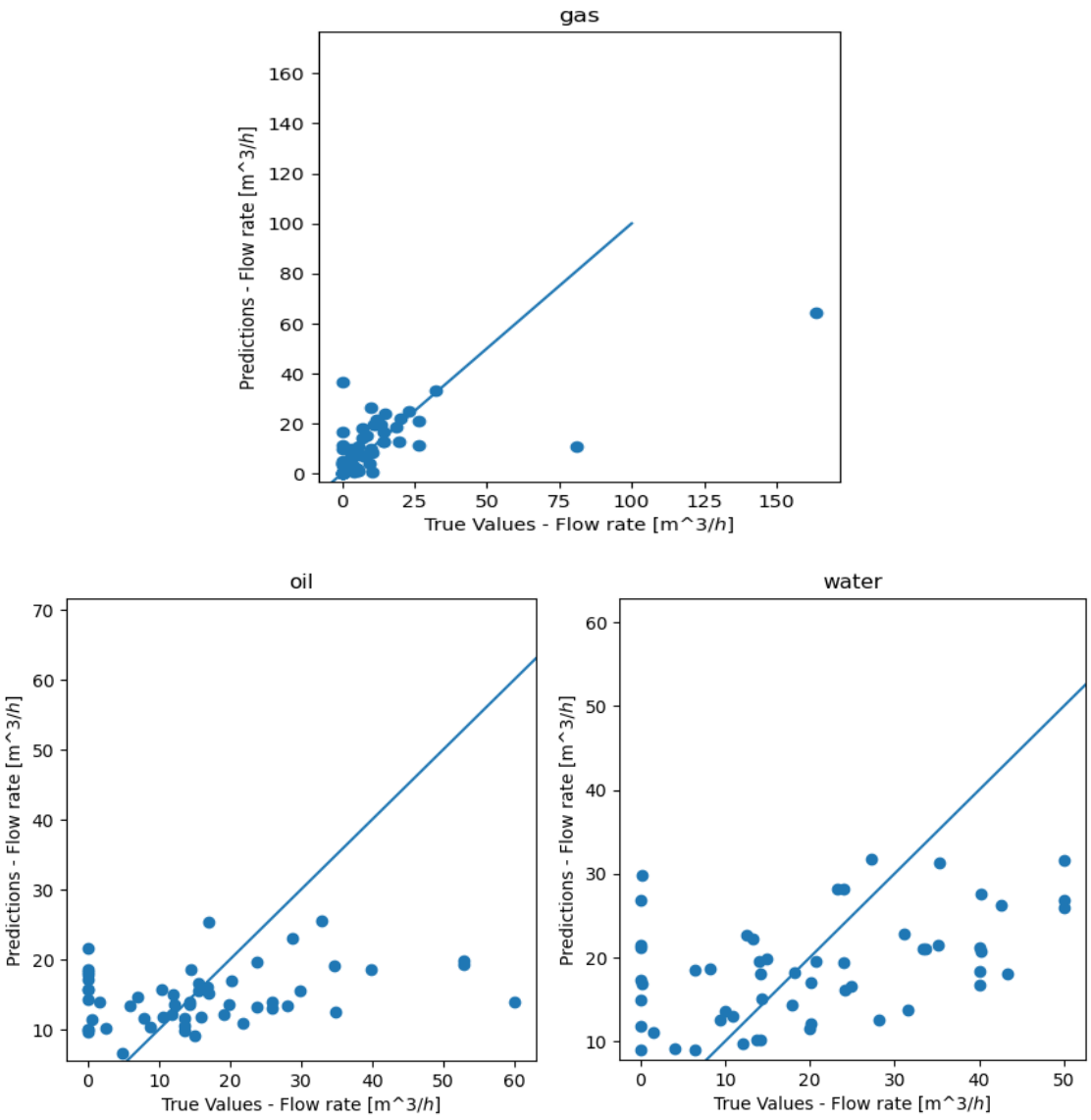

Figure 34: Regression plots - multiphase flow prediction model trained on three accelerometer data.

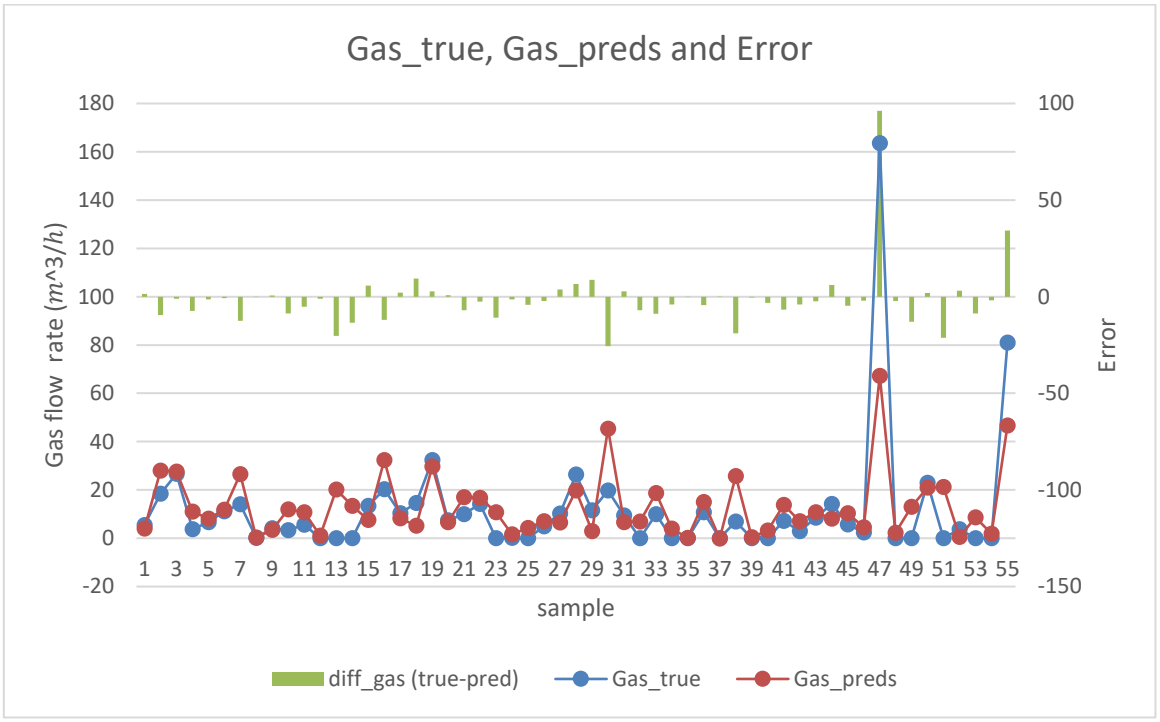

Figure 35: Gas true values and predicted values comparison – Multiphase flow (three accelerometer) test data.

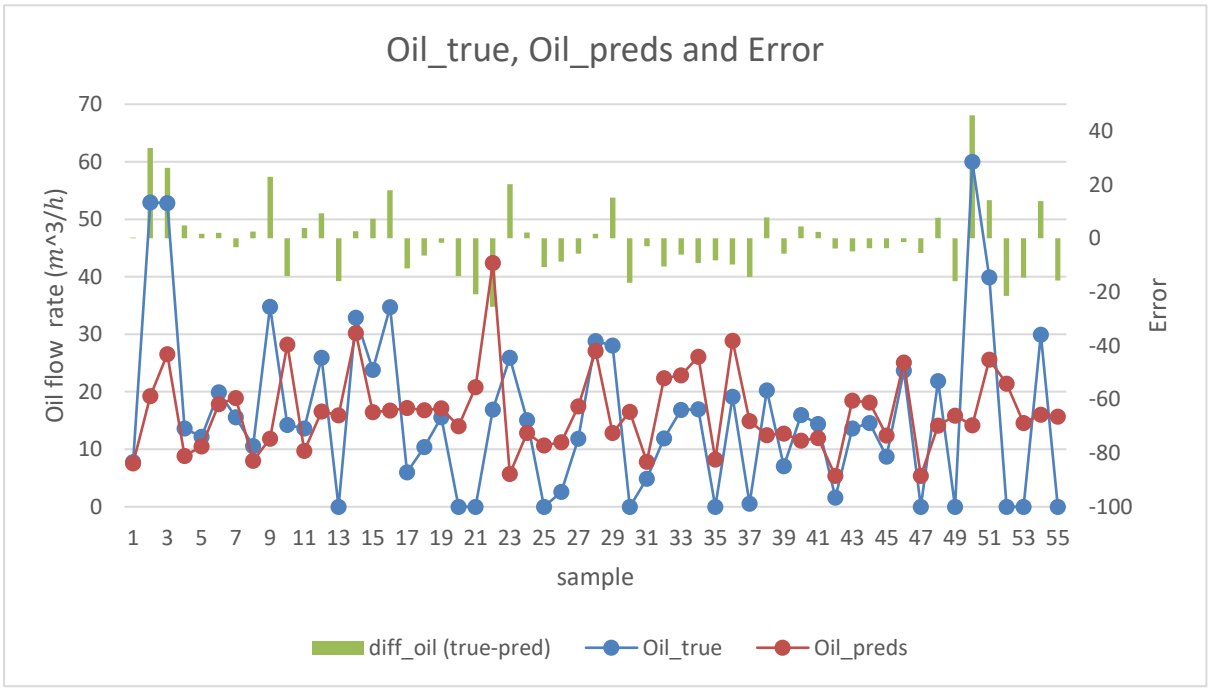

Figure 36: Oil true values and predicted values comparison – Multiphase flow (three accelerometer) test data.

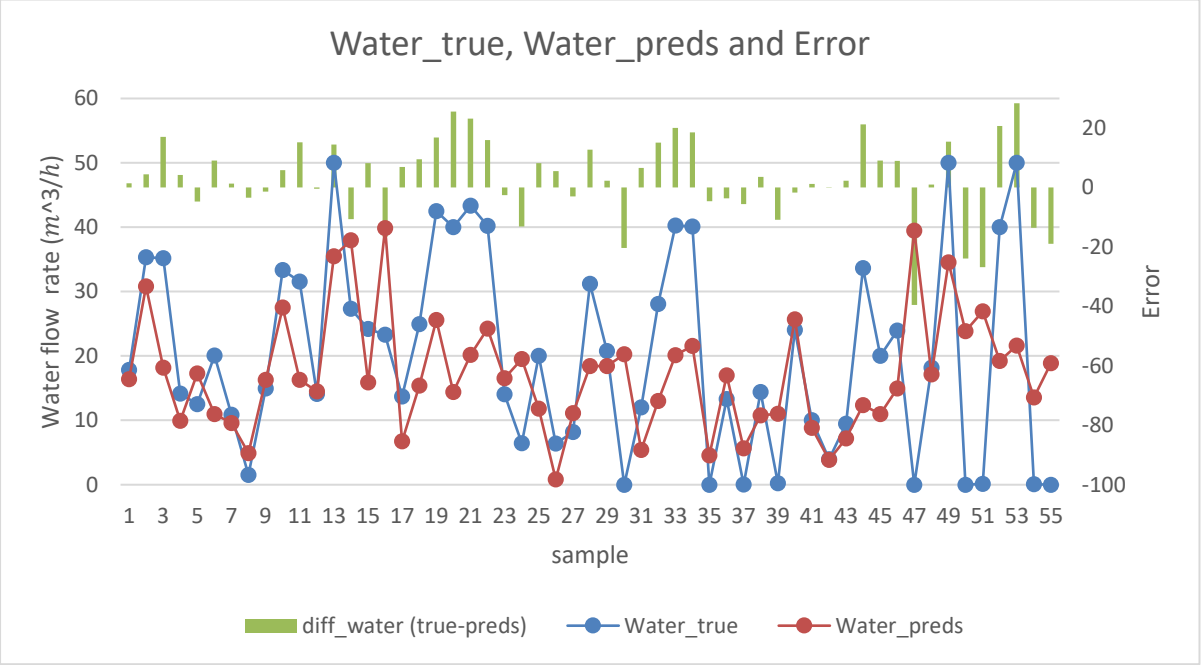

Figure 37: Water true values and predicted values comparison – Multiphase flow (three accelerometer) test data.

7.3 Results of predicting single-phase flow rate using three accelerometers

Table 10 shows RMSE values on test data of single-phase flow rate prediction model trained on three accelerometer data are worse than the model trained on 4 accelerometer data.

In Figure 38, it can be observed from gas, oil and water flow rate prediction regression plots, that the single-phase model has improved performance in single-phase flow samples than in multiphase flow rate prediction model trained on three accelerometer data.

Observing Figure 39, Figure 40 and Figure 41, It is clear that single-phase flow prediction model trained with three accelerometer data is not as good as the one trained with four accelerometer data; while both having the same dataset to train with. But still model trained with three accelerometer data has been able to find patterns in the data and make good predictions.

Appendix E contains the true values and predictions with prediction errors from single-phase flow rate prediction model trained with three accelerometer data.

Table 10: RMSE values from test data – Single-phase flow model (three accelerometers).

Gas RMSE $[m^3/h]$	33.39
Oil RMSE $[m^3/h]$	7.67
Water RMSE $[m^3/h]$	17.98

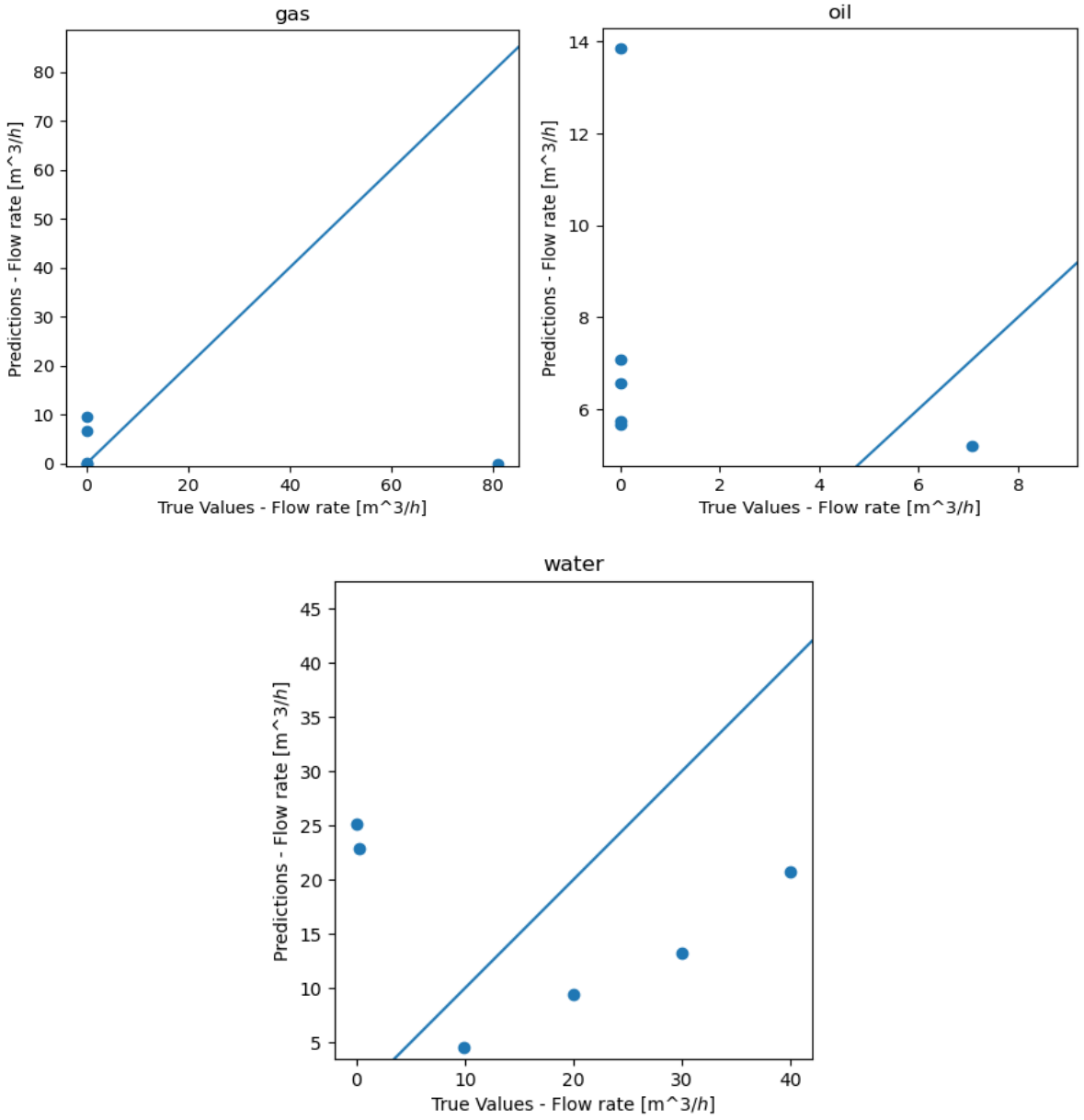

Figure 38: Regression plots - single-phase flow rate prediction model trained on 3 accelerometer data.

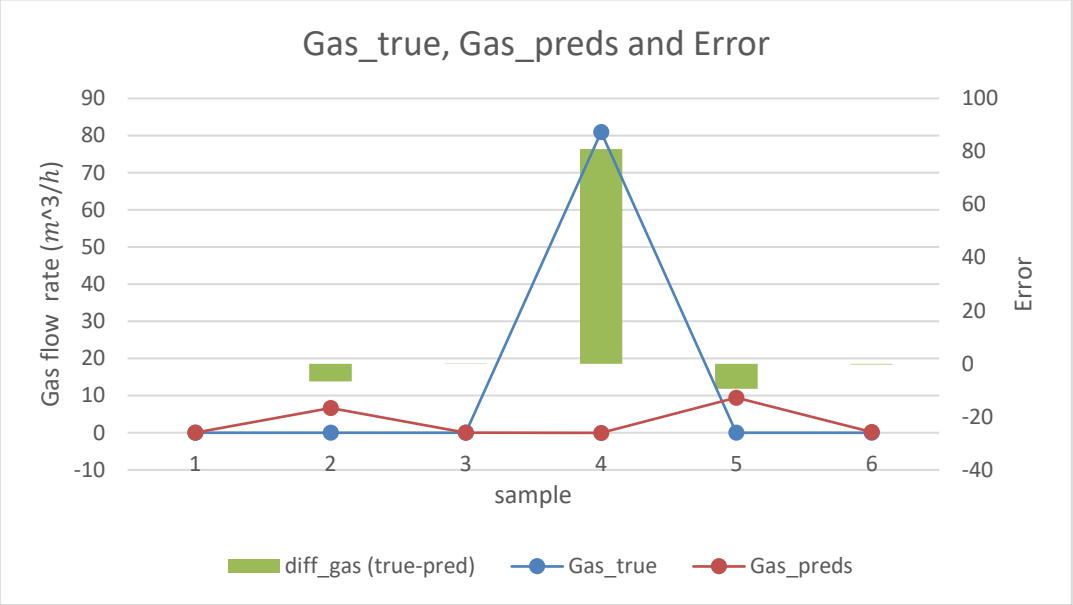

Figure 39: Gas true values and predicted values comparison – Single-phase flow (three accelerometer) test data.

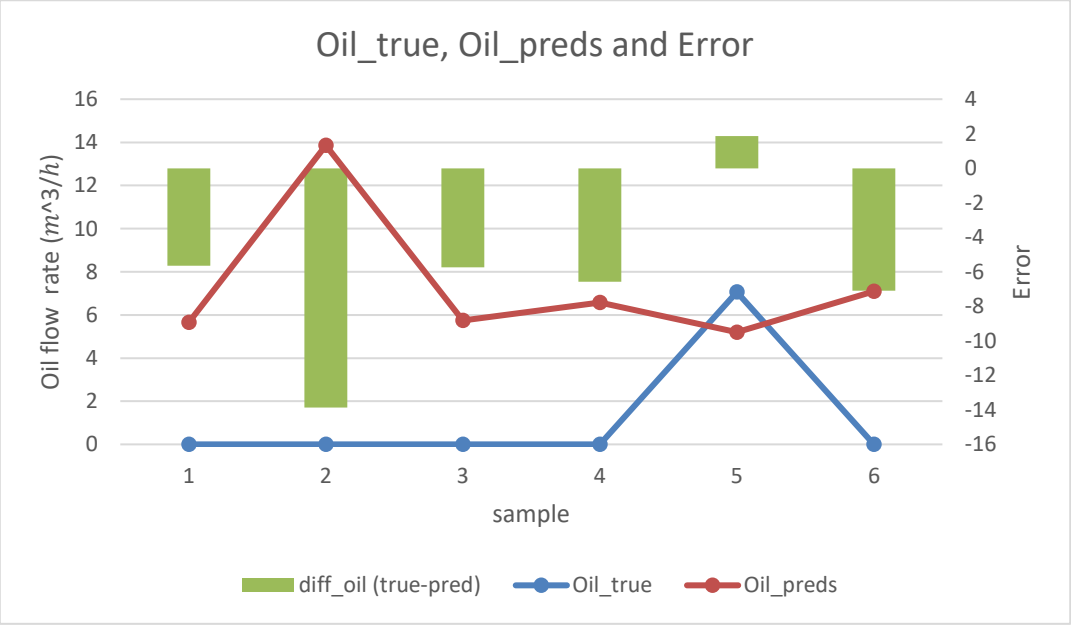

Figure 40: Oil true values and predicted values comparison – Single-phase flow (three accelerometer) test data.

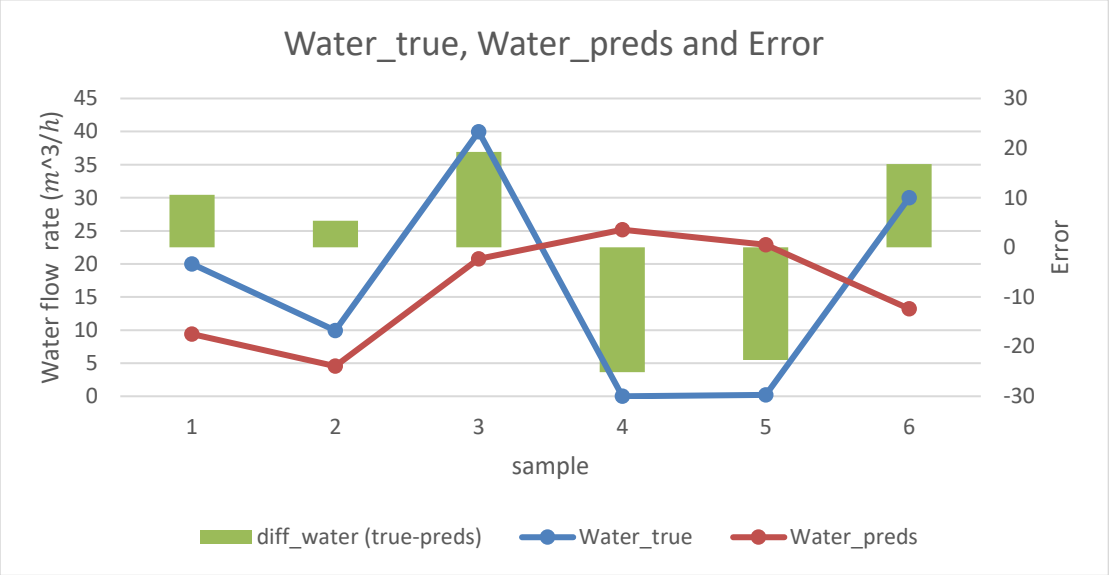

Figure 41: Water true values and predicted values comparison – Single-phase flow (three accelerometer) test data.

It can be concluded that training models with three accelerometer data or less could be sufficient considering the availability of more samples as models trained with four accelerometer data are making better predictions.

7.4 Results of flow type identification

The CNN model that has been trained on the generated spectrogram data in has made classification of flow types with 68.75% accuracy.

Figure 42 is the confusion matrix of the CNN image recognition flow type detection model. The confusion is created based on the prediction of the model on test data. Among the 7 classes, in test dataset – 'G' has 2 samples, 1 of which is correctly classified correctly and 1 is misclassified as 'O'. Of the 2 samples of 'GO', 1 is correctly classified as 'GO', and 1 misclassified as 'GOW'. Of the 3 samples of 'GOW' 2 is correctly classified and 1 is misclassified as 'O'. 1 of the 2 'GW' samples is correctly classified as "GW" and 1 is misclassified as 'GO'. 1 'O' is classified correctly and 1 is misclassified as 'OW'. All 3 samples of 'OW' class have been classified correctly and all 2 samples of 'W' class have been classified correctly.

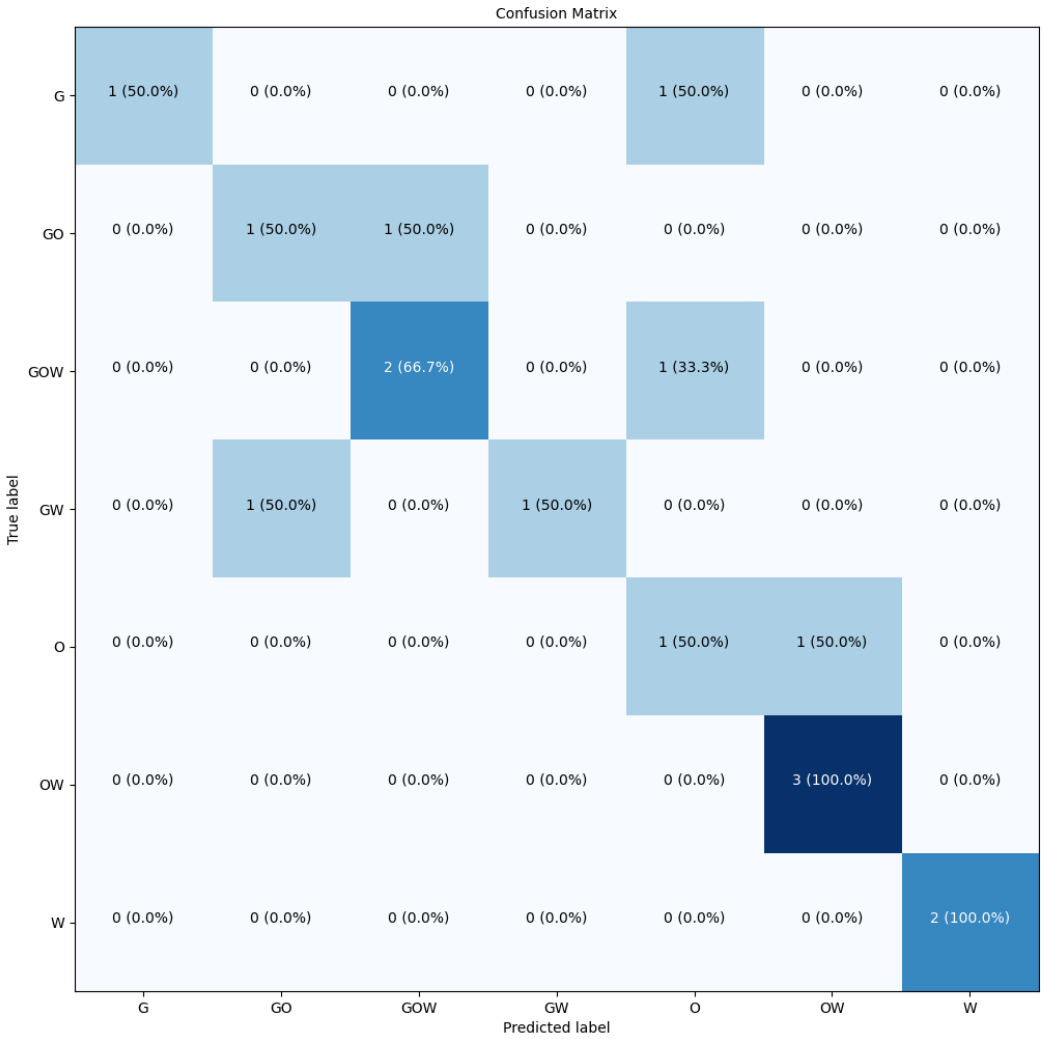

Figure 42: Confusion matrix from CNN flow type detection model. The vertical axis represents the true labels, and the horizontal axis represents the predicted labels of flow type classification. Cells with a higher colour intensity indicate a larger number of samples, while cells with a lower intensity represent a smaller number of samples.

Table 11 shows the analysis of image recognition model's performance by comparing true labels and classified label by the model. It the confidence of every prediction as well.

Table 11: True labels and predicted labels analysis.

Prediction confidence	True label	Predicted label	Prediction correct?
0.48	G	O	FALSE
0.69	G	G	TRUE
0.63	GO	GOW	FALSE
0.41	GO	GO	TRUE
0.61	GOW	GOW	TRUE
0.54	GOW	O	FALSE
0.88	GOW	GOW	TRUE
0.41	GW	GO	FALSE
0.43	GW	GW	TRUE
0.42	O	O	TRUE
0.37	O	OW	FALSE
0.33	OW	OW	TRUE
0.44	OW	OW	TRUE
0.68	OW	OW	TRUE
0.38	W	W	TRUE
0.23	W	W	TRUE

The accuracy of the model is quite good considering that the model has been trained on very low number of samples.

7.5 Results of physics-based multiphase flow rate prediction

Table 6 shows the RMSE values of physics-based multiphase flow model predictions. The RMSE values that have been achieved due to the custom loss function are almost as good as the general ANN multiphase flow rate prediction model RMSE values in chapter 7.1.

Figure 43 displays regression plots of physics-based multiphase flow model predictions. It can be observed that the regression plots of physics-based ANN model is quite similar to usual ANN models trained previously. This model is unable to predict instances where the true value is zero as well.

In Figure 44, Figure 45, Figure 46 true flow rates and predicted flow rates of gas, oil and water from physics-based multiphase flow model can be observed. The predictions are almost as good as the general ANN multiphase flow prediction model's predictions.

Appendix F contains the true values and predictions with prediction errors from physics-based multiphase flow rate prediction model.

Table 12: RMSE values from test data – Physics-based multiphase flow model

Gas RMSE $[m^3/h]$	16.87
Oil RMSE $[m^3/h]$	13.76
Water RMSE $[m^3/h]$	13.85

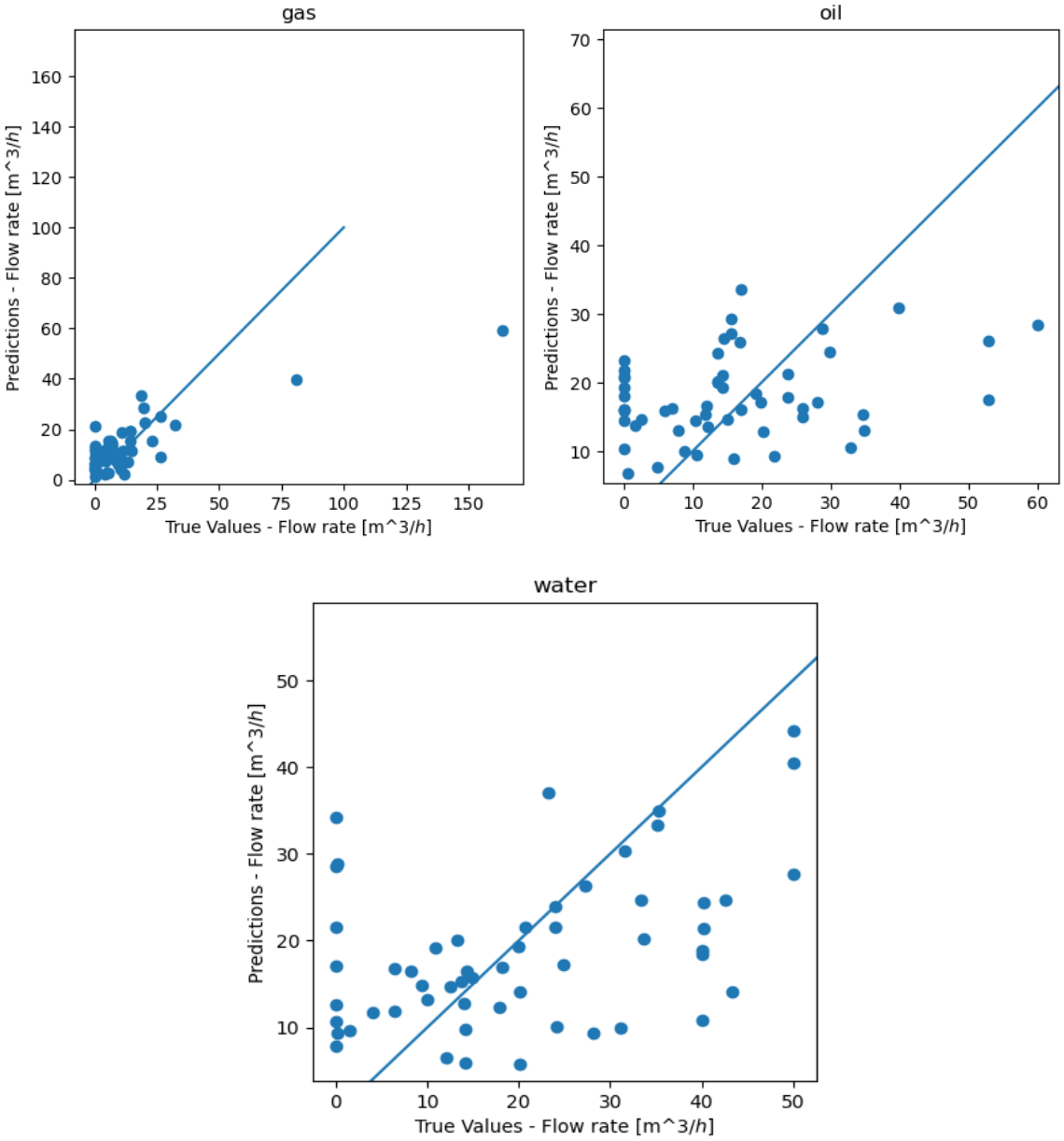

Figure 43: Regression plots physics-based multiphase flow predictions.

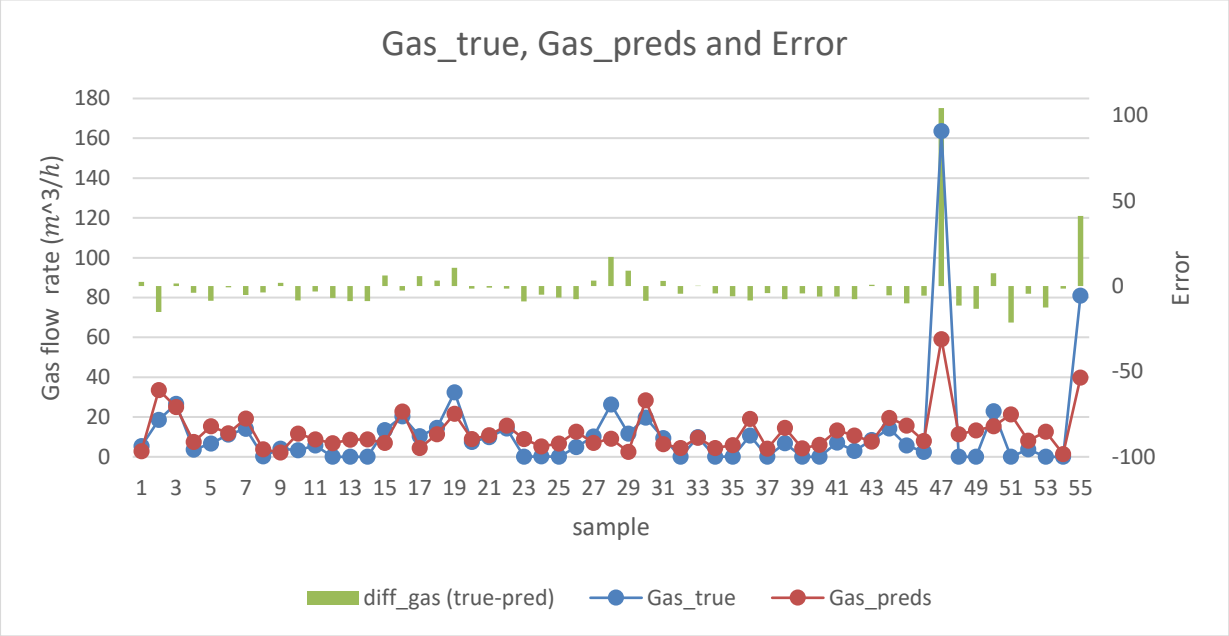

Figure 44: Gas true values and predicted values comparison – Physics-based Multiphase flow - test dataset.

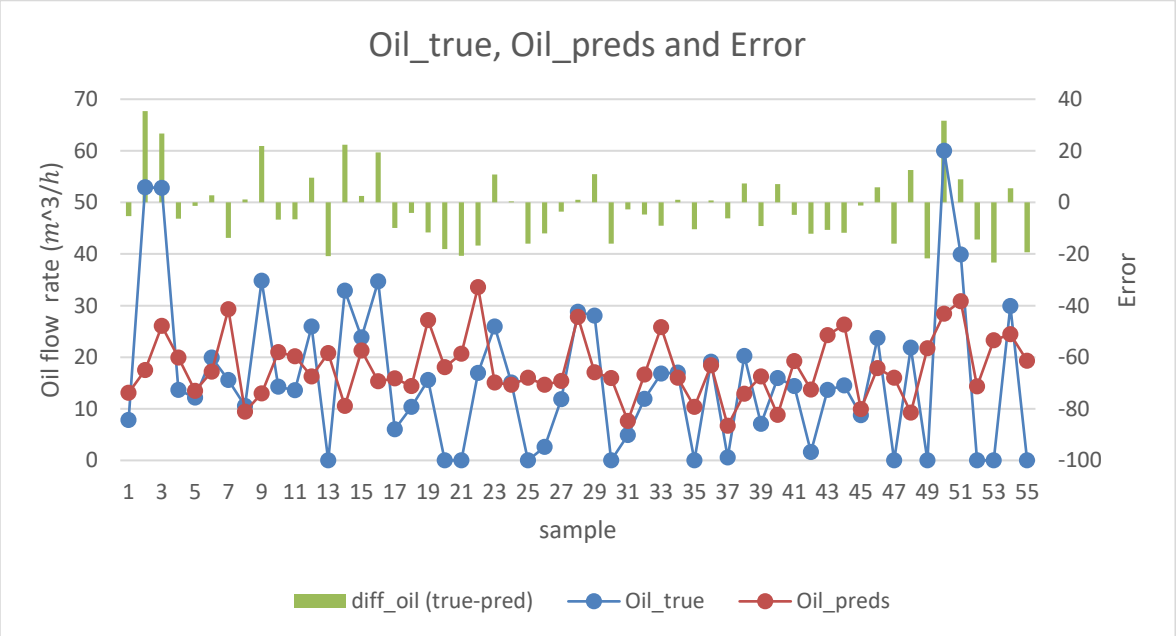

Figure 45: Oil true values and predicted values comparison – Physics-based Multiphase flow - test dataset

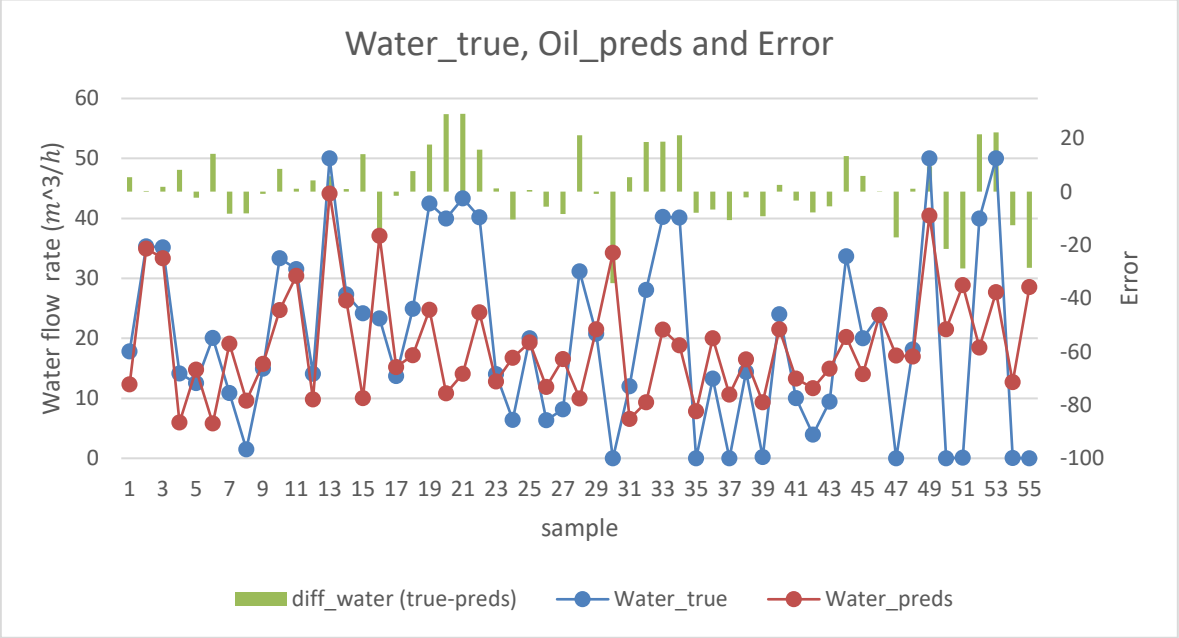

Figure 46: Water true values and predicted values comparison – Physics-based Multiphase flow - test dataset.

.

8 Discussion

In this chapter, the outcomes of the thesis, and possible ways of improving the outcomes are discussed.

8.1 Denoising of data from Coriolis meter noise

The denoising process that has been applied in this research may not be an infallible solution. The applied STFT denoising process is usually applied in audio data background noise reduction process. In the case of audio data, it is possible to listen to the audio and check the quality of the denoising result. This method is known as subjective evaluation of audio data, [40]. Human listeners can perform subjective evaluation after denoising audio data by rating the speech clarity, musicality and presence artifacts or distortion; thus, modifications can be made if required in the denoising process accordingly. But in this research, it was not known how the Coriolis meter noise signal looks like or what are the properties of the noise. The only information that was available is that the Coriolis meter creates a vibration of around 200 Hz, and it is of low amplitude. These two characteristics have been kept in mind during the work on denoising process. Thus, it can be said that the denoising process applied in this project is a tentative denoising process. It is necessary that a reliable and accurate denoising process is developed for such cases in future.

The ideal scenario for this research would be to generate the same kind of data after excluding the Coriolis meter from the Equinor test rig or collecting data from a rig that does not create such noise issues. It is hoped that it would significantly improve the results of flow rate prediction with accelerometer data only and flow type classification as well.

8.2 Flow rate prediction

Observing the results of flow rate prediction models, the accelerometer data contains the necessary information to predict flow rate of oil, gas and water. The research also proves that the models can predict flow rate even with fewer than all of four accelerometers. However, it is necessary that the model has more samples to train on, especially for single-phase samples.

The neural network structures underwent extensive tuning via numerous configurations. There could be other configurations with a different number of neurons and different activation functions in different layers that could improve the result. If more samples are added to the data set, then the neural network model, training duration and batch size should require changes accordingly.

In practical applications, if adding more features such pressure data, temperature data etc. is feasible, then they could be added besides accelerometer data which could make it possible to predict flow rate with low RMSE in case of having low number of samples. If it is a requirement

to use only accelerometer data, then abundant samples should be recorded for model training purposes.

8.3 Flow type classification

Due to unbalanced classes, many samples had to be discarded. The image recognition model has been trained on 5 to 12 samples from each class. 68.75% accuracy is still quite satisfactory considering the low number of samples used for training. Having few samples for validation also affects the training process of a model negatively. But the accuracy achieved proves clearly that if more samples are included in the training process, the model will perform better with flying colors.

The backbone model has been kept frozen, meaning the layers inside the backbone model have not been trained. They performed in this experiment with their default trained weights. Usually, when there are more than 1000 samples available for each class, some layers of the backbone model are unfrozen and keeping the training rate very low, the unfrozen layers are trained. This process is called fine-tuning, [41], [42]. Fine-tuning could be applied in the flow type classification process if there are at least 1000 samples available in each class.

The spectrograms could turn out to be better if Coriolis meter noise was not present in the data. It can be assumed that the Coriolis meter noise has somewhat affected the flow type identification process negatively.

8.4 Physics-based or informed flow rate prediction

The aim of creating physics-based flow rate prediction model in this research has only been to explore the possibilities and understand the behavior of neural networks attempting to predict flow rate while physics is involved within the codes. What has been observed is that it became quite easier to create a similar result as before while a non-physics informed model was being tuned. During the training of the non-physics informed models, it took many more attempts to merge the training and validation curves. And the model was quite unstable concerning slight changes to the configurations. Small change would have had a huge impact on the results. Which made it difficult to attain the result that has been achieved at the end. On the contrary, while working with the physics-informed model, it was quite stable, and it was easier to assume how a change affected the model's results. Thus, tuning it to the result that has been achieved became quite easier. But observing the predictions made by both non-physics based or informed model and physics-based model, it can be said that it was not possible to make the results of physics-based model exactly as good as the non-physics model.

9 Conclusion

This thesis contributes to developing machine learning techniques for flow rate prediction and flow type classification.

Based on the results and discussion presented in this thesis, it can be concluded that the use of accelerometer data in combination with machine learning techniques shows promise in predicting flow rate and classifying flow types. The denoising process applied in this research was tentative, and there is a need for a reliable and accurate denoising process for Coriolis meter noise in future studies. Adding more features such as pressure and temperature data could improve the accuracy of flow rate prediction models, especially with low numbers of samples.

The intriguing findings of the analysis on flow rate prediction reveal that accelerometer data houses the indispensable information to predict the flow rate of oil, gas, and water, and the models can accomplish such prognostications with fewer than four accelerometers. Nevertheless, it is crucial to keep in mind that the models' proficiency is inextricably linked to the quantity of samples utilized for training, and to guarantee optimal performance, a larger corpus of samples is needed, particularly for single-phase samples. Furthermore, for the classification of flow type, it might be worthwhile to explore the possibility of fine-tuning the models if an abundant quantity of samples is obtainable in each category.

Physics-based models in flow rate prediction has shown potential in creating more stable models that are easier to tune compared to non-physics informed models. Unfortunately, while the potential benefits are undeniable, the actual results of the physics-based model have been somewhat underwhelming, failing to match the performance of the non-physics model.

9.1 Suggested future work

The findings suggest that future studies should focus on developing a reliable and accurate denoising process for Coriolis meter noise. Collecting more samples for training is necessary in achieving great results using only accelerometer data for flow rate prediction and flow type identification. The number of recorded samples should be kept in balance regarding all types of flow. Further exploration in the use of physics-based models in combination with machine learning techniques is highly recommended as the use of physics has shown some positive effects during tuning of the physics-based model using custom loss created based on Darcy-Weisbach head loss equation.

References

[1] C. E. Brennen, *Fundamentals of multiphase flow*. Cambridge [England] ; New York: Cambridge University Press, 2005.

[2] Z. Jiang, H. Wang, Y. Yang, and Y. Li, "Comparison of machine learning methods for multiphase flowrate prediction," in *2019 IEEE International Conference on Imaging Systems and Techniques (IST)*, Abu Dhabi, United Arab Emirates: IEEE, Dec. 2019, pp. 1–6. doi: 10.1109/IST48021.2019.9010450.

[3] R. Yan, H. Viumdal, K. Fjalestad, and S. Mylvaganam, "Ensemble learning in the estimation of flow types and velocities of individual phases in multiphase flow using non-intrusive accelerometers' and process pressure data," in *2022 IEEE Sensors Applications Symposium (SAS)*, Aug. 2022, pp. 1–6. doi: 10.1109/SAS54819.2022.9881352.

[4] Y. Zhang, A. N. Azman, K.-W. Xu, C. Kang, and H.-B. Kim, "Two-phase flow regime identification based on the liquid-phase velocity information and machine learning," *Exp. Fluids*, vol. 61, no. 10, p. 212, Oct. 2020, doi: 10.1007/s00348-020-03046-x.

[5] L. S. Hansen, S. Pedersen, and P. Durdevic, "Multi-Phase Flow Metering in Offshore Oil and Gas Transportation Pipelines: Trends and Perspectives," *Sensors*, vol. 19, no. 9, p. 2184, May 2019, doi: 10.3390/s19092184.

[6] W. K. H. Ariyaratne, E. V. P. J. Manjula, C. Ratnayake, and M. C. Melaaen, "CFD Approaches for Modeling Gas-Solids Multiphase Flows - A Review," presented at the Proceedings of The 9th EUROSIM Congress on Modelling and Simulation, EUROSIM 2016, The 57th SIMS Conference on Simulation and Modelling SIMS 2016, Dec. 2018, pp. 680–686. doi: 10.3384/ecp17142680.

[7] M. Du, H. Yin, X. Chen, and X. Wang, "Oil-in-Water Two-Phase Flow Pattern Identification From Experimental Snapshots Using Convolutional Neural Network," *IEEE Access*, vol. 7, pp. 6219–6225, 2019, doi: 10.1109/ACCESS.2018.2888733.

[8] D. E. Turney, D. V. Kalaga, M. Ansari, R. Yakobov, and J. B. Joshi, "Reform of the drift-flux model of multiphase flow in pipes, wellbores, and reactor vessels," *Chem. Eng. Sci.*, vol. 184, pp. 251–258, Jul. 2018, doi: 10.1016/j.ces.2018.03.033.

[9] T. Raja Sekhar and P. Satapathy, "Group classification for isothermal drift flux model of two phase flows," *Comput. Math. Appl.*, vol. 72, no. 5, pp. 1436–1443, Sep. 2016, doi: 10.1016/j.camwa.2016.07.017.

[10] T. Bikmukhametov, "Machine Learning and First Principles Modeling Applied to Multiphase Flow Estimation".

[11] C. Chao, X. Xu, S. O. Kwelle, and X. Fan, "Significance of gas-liquid interfaces for two-phase flows in micro-channels," *Chem. Eng. Sci.*, vol. 192, pp. 114–125, Dec. 2018, doi: 10.1016/j.ces.2018.07.026.

[12] J. Hapanowicz and L. Troniewski, "Two-phase flow of liquid–liquid mixture in the range of the water droplet pattern," *Chem. Eng. Process. Process Intensif.*, vol. 41, no. 2, pp. 165–172, Feb. 2002, doi: 10.1016/S0255-2701(01)00127-1.

[13] M. Mortazavi, “Two-phase flow pressure drop in PEM fuel cell flow channel bends,” *Int. J. Multiph. Flow*, vol. 143, p. 103759, Oct. 2021, doi: 10.1016/j.ijmultiphaseflow.2021.103759.

[14] J. Buist, B. Sanderse, Y. van Halder, B. Koren, and G. van Heijst, “MACHINE LEARNING FOR CLOSURE MODELS IN MULTIPHASE FLOW APPLICATIONS,” in *Proceedings of the 3rd International Conference on Uncertainty Quantification in Computational Sciences and Engineering (UNCECOMP 2019)*, Crete, Greece: Institute of Structural Analysis and Antiseismic Research School of Civil Engineering National Technical University of Athens (NTUA) Greece, 2019, pp. 379–399. doi: 10.7712/120219.6348.18409.

[15] “How to Measure Acceleration?,” *https://www.omega.com/en-us/*. https://www.omega.com/en-us/resources/accelerometers (accessed Jan. 26, 2023).

[16] K. A. R. Medeiros, C. R. H. Barbosa, and E. C. de Oliveira, “Flow Measurement by Piezoelectric Accelerometers: Application in the Oil Industry,” *Pet. Sci. Technol.*, vol. 33, no. 13–14, pp. 1402–1409, Jul. 2015, doi: 10.1080/10916466.2015.1044613.

[17] “What are Neural Networks? | IBM.” https://www.ibm.com/topics/neural-networks (accessed Jan. 26, 2023).

[18] M. Al-Naser, M. Elshafei, and A. Al-Sarkhi, “Artificial neural network application for multiphase flow patterns detection: A new approach,” *J. Pet. Sci. Eng.*, vol. 145, pp. 548–564, Sep. 2016, doi: 10.1016/j.petrol.2016.06.029.

[19] S. S. Liew, “An Efficient and Effective Convolutional Neural Network for Visual Pattern Recognition,” 2016. doi: 10.13140/RG.2.2.21172.86405.

[20] “Convolutional Neural Network Tutorial [Update],” *Simplilearn.com*. https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network (accessed Apr. 30, 2023).

[21] P. ray, “Convolutional Neural Network (CNN) and its Application- All u need to know,” *Analytics Vidhya*, Jan. 19, 2021. https://medium.com/analytics-vidhya/convolutional-neural-network-cnn-and-its-application-all-u-need-to-know-f29c1d51b3e5 (accessed Apr. 30, 2023).

[22] H. S. Chatterjee, “Various Types of Convolutional Neural Network,” *Medium*, Jul. 25, 2022. https://towardsdatascience.com/various-types-of-convolutional-neural-network-8b00c9a08a1b (accessed Apr. 30, 2023).

[23] “Coriolis Flow Meter Principles | Emerson US.” https://www.emerson.com/en-us/automation/measurement-instrumentation/flow-measurement/coriolis-flow-meters (accessed Jan. 19, 2023).

[24] “What is a Coriolis Flow Meter and How Does it Work?,” *https://www.omega.com/en-us/*. https://www.omega.com/en-us/resources/what-is-a-coriolis-flow-meter (accessed Jan. 19, 2023).

[25] “Short-time Fourier transform - MATLAB stft.” https://www.mathworks.com/help/signal/ref/stft.html (accessed Feb. 15, 2023).

[26] “Short-time Fourier transform,” *Wikipedia*. Oct. 11, 2022. Accessed: Feb. 15, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Short-time_Fourier_transform&oldid=1115436538

[27] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, “Physics-informed machine learning,” *Nat. Rev. Phys.*, vol. 3, no. 6, Art. no. 6, Jun. 2021, doi: 10.1038/s42254-021-00314-5.

[28] N. Thuerey, P. Holl, M. Mueller, P. Schnell, F. Trost, and K. Um, “Physics-based Deep Learning.” arXiv, Apr. 25, 2022. doi: 10.48550/arXiv.2109.05237.

[29] “Measurements and Feature Extraction - MATLAB & Simulink - MathWorks India.” https://in.mathworks.com/help/signal/measurements-and-feature-extraction.html (accessed May 15, 2023).

[30] D. Mwiti, “Transfer Learning Guide: A Practical Tutorial With Examples for Images and Text in Keras,” *neptune.ai*, Jul. 21, 2022. https://neptune.ai/blog/transfer-learning-guide-examples-for-images-and-text-in-keras (accessed May 01, 2023).

[31] “Pretrained Deep Neural Networks - MATLAB & Simulink.” https://www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html (accessed May 01, 2023).

[32] “TensorFlow,” *Wikipedia*. Apr. 20, 2023. Accessed: May 01, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=TensorFlow&oldid=1150793006

[33] P. Yakubovskiy, “efficientnet: EfficientNet model re-implementation. Keras and TensorFlow Keras.” Accessed: May 01, 2023. [Online]. Available: https://github.com/qubvel/efficientnet

[34] M. Tan and Q. V. Le, “EfficientNetV2: Smaller Models and Faster Training.” arXiv, Jun. 23, 2021. doi: 10.48550/arXiv.2104.00298.

[35] K. Team, “Keras documentation: Image classification via fine-tuning with EfficientNet.” https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/ (accessed May 01, 2023).

[36] K. Team, “Keras documentation: EfficientNet B0 to B7.” https://keras.io/api/applications/efficientnet/ (accessed May 01, 2023).

[37] “Darcy–Weisbach equation,” *Wikipedia*. Mar. 26, 2023. Accessed: Apr. 26, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Darcy%E2%80%93Weisbach_equation&oldid=1146657509

[38] G. O. Brown, “The History of the Darcy-Weisbach Equation for Pipe Flow Resistance,” in *Environmental and Water Resources History*, Washington, D.C., United States: American Society of Civil Engineers, Oct. 2002, pp. 34–43. doi: 10.1061/40650(2003)4.

[39] C. Y. P.E, “Darcy-Weisbach Equation for Calculating Pressure Losses,” *EngineerExcel*, Jan. 11, 2022. https://engineerexcel.com/darcy-weisbach-equation-for-calculating-pressure-losses/ (accessed Jan. 15, 2023).

[40] E. Vincent, M. Jafari, and M. Plumbley, “Preliminary guidelines for subjective evaluation of audio source separation algorithms,” Apr. 2012.

[41] joseph, “Fine Tuning in Deep Learning: What You Need to Know - reason.town,” Aug. 15, 2022. https://reason.town/fine-tuning-in-deep-learning/ (accessed May 14, 2023).

[42] V. Roman, “CNN Transfer Learning & Fine Tuning,” *Medium*, Apr. 17, 2021. https://towardsdatascience.com/cnn-transfer-learning-fine-tuning-9f3e7c5806b2 (accessed May 14, 2023).

Appendices

Appendix A: Task Description

Appendix B: Gas, Oil and Water true value and predicted values comparison - Multiphase flow test data

Appendix C: True values and predicted values comparison - test dataset – single-phase

Appendix D: True values and predicted values comparison - test dataset – Multiphase model (three accelerometer)

Appendix E: True values and predicted values comparison - test dataset – Single-phase model (three accelerometer)

Appendix F: True values and predicted values comparison - test dataset – Physics-based multiphase model

Appendix G: Code for feature extraction

Appendix H: Code for multiphase flow Neural Network

Appendix I: Code for single-phase flow prediction neural network

Appendix J: Code for multiphase flow prediction neural network (trained with three accelerometer data)

Appendix K: Code for single-phase flow rate prediction neural network (trained with three accelerometer data

Appendix L: Code for spectrogram generation automated scriptAppendix M: Code for splitting spectrogram dataset into train and test set

Appendix M: Code for splitting spectrogram dataset into train and test set

Appendix N: Code for CNN image recognition model

Appendix O: Code for physics-based multiphase flow rate prediction model with customized loss function

Appendix A: Task Description

University of South-Eastern Norway

Faculty of Technology, Natural Sciences and Maritime Sciences, Campus Porsgrunn

FMH606 Master's Thesis

Title: Velocity monitoring of phases in oil/gas/water multiphase flow

USN supervisor: Ru Yan; Saba Mylvaganam

External partner: Kjetil Fjalestad/EQUINOR/, Tonni Franke Johansen/ SINTEF

Task background:
Multiphase flow rigs in USN, Campus Porsgrunn and EQUINOR in Herøya, Grenland have been used in multitude of experiments involving bachelor, master, and PhD candidates from Norwegian and International students and researchers, resulting in many student projects and research publications. Using AE-sensors, accelerometers, and conventional measurements such as temperature, pressure, flow, and absorption of gamma rays, tomometric measurements using electrical resistance and capacitance tomographic equipment have been used in these multiphase flow measurements.

This project aims to get an overview of the achieved results with focus on velocity measurements of the phases in multiphase flow involving oil, gas and water and improve the performance of algorithms used in the previous studies. The focus is on data from non-intrusive and non-invasive sensors such as AE-sensors and accelerometers and exploitation of physics aided ML/AI techniques.

Task description:
The tentative list of tasks for this thesis work is as follows:
(1) Comprehensive and thorough survey of the work done in multiphase flowmetering using the USN and EQUINOR multiphase flow rigs.
(2) Coupled to task (1), a short survey of the contemporary publications related to the same topics
(3) A comprehensive overview of the algorithms used and their performances with respect to repeatability, uncertainty, and processing time.
(4) Use existing data from multiphase flow experiments done in USN and EQUINOR and test algorithms with the necessary modifications to estimate the phase velocities.
(5) Developing new ML/AI models based on already existing models in the estimation of phase velocities
(6) Include a physics based ML/AI algorithm for the estimation of single phase flow
(7) Submitting a Master Thesis following the guidelines of USN with necessary programs and including a well-documented and complete set of all experimental data from the measurements

Student category: IIA students

The task is suitable for online students (not present at the campus): Yes

Practical arrangements:
Necessary experimental data will be provided by USN and possibly EQUINOR. This work is closely coupled to an ongoing project SAM (SAM Self Adapting Model-based system for Process Autonomy - SINTEF) and may involve partners in the project Digipro.

Supervision:
As a general rule, the student is entitled to 15-20 hours of supervision. This includes necessary time for the supervisor to prepare for supervision meetings (reading material to be discussed, etc.).

Signatures:

Supervisor (date and signature): 01.02.2023

Student (write clearly in all capitalized letters): ABDUR RAHMAN

Student (date and signature): 02/04/2023 A. Rahman

Appendix B: Gas, Oil and Water true value and predicted values comparison - Multiphase flow test data

	Gas true	Gas preds	Gas error	Oil true	Oil preds	Oil error	Water true	Water preds	Water error
0	5.45	4.01	1.44	7.8	10.42	-2.62	17.81	9.11	8.7
1	18.49	27.94	-9.45	52.93	23.27	29.66	35.32	26.9	8.42
2	26.62	27.6	-0.98	52.8	27.94	24.86	35.2	22.47	12.73
3	3.68	10.96	-7.28	13.63	16	-2.37	14.16	11.74	2.42
4	6.71	8.14	-1.43	12.16	11.12	1.04	12.53	15.93	-3.4
5	11.13	11.88	-0.75	19.92	18.98	0.94	20.09	9.08	11.01
6	14.09	26.51	-12.42	15.58	32.03	-16.45	10.89	16.53	-5.64
7	0.16	0.3	-0.14	10.57	13.3	-2.73	1.5	5.47	-3.97
8	4.07	3.45	0.62	34.8	15.61	19.19	14.95	19.11	-4.16
9	3.29	12	-8.71	14.25	21.94	-7.69	33.35	20.8	12.55
10	5.64	10.79	-5.15	13.61	19.65	-6.04	31.56	18.01	13.55
11	0	1.03	-1.03	25.91	16.41	9.5	14.1	13.67	0.43
12	0	20.19	-20.19	0	23.47	-23.47	50	32.42	17.58
13	0	13.45	-13.45	32.91	24.7	8.21	27.33	30.41	-3.08
14	13.4	7.62	5.78	23.8	19.27	4.53	24.18	17.84	6.34
15	20.27	32.31	-12.04	34.7	23.79	10.91	23.31	36.55	-13.24
16	10.34	8.29	2.05	6	14.53	-8.53	13.71	10.54	3.17
17	14.58	5.2	9.38	10.37	16.63	-6.26	24.93	20.17	4.76
18	32.37	29.67	2.7	15.57	25.79	-10.22	42.52	27.28	15.24
19	7.41	6.62	0.79	0	16.79	-16.79	40	12.13	27.87
20	9.93	16.92	-6.99	0	23.17	-23.17	43.34	21.63	21.71
21	14.24	16.74	-2.5	16.92	27.83	-10.91	40.19	23.16	17.03
22	0	10.75	-10.75	25.92	10.97	14.95	14.06	15.71	-1.65
23	0.18	1.57	-1.39	15.05	10.31	4.74	6.42	20.67	-14.25
24	0	4.26	-4.26	0	13.79	-13.79	20	21.24	-1.24
25	4.92	7.09	-2.17	2.6	6.38	-3.78	6.37	3.41	2.96
26	10.22	6.46	3.76	11.83	17.89	-6.06	8.18	10.25	-2.07
27	26.34	19.83	6.51	28.8	27.59	1.21	31.19	22.85	8.34
28	11.6	2.89	8.71	28.07	22.35	5.72	20.77	24.41	-3.64
29	19.75	45.29	-25.54	0	10.41	-10.41	0	31.38	-31.38
30	9.43	6.62	2.81	4.9	5.13	-0.23	12.01	7.69	4.32
31	0	6.97	-6.97	11.91	22.39	-10.48	28.09	9.96	18.13

32	9.9	18.77	-8.87	16.84	28.08	-11.24	40.27	21.08	19.19
33	0.05	4.05	-4	16.98	23.47	-6.49	40.12	25.33	14.79
34	0.06	0.12	-0.06	0	8.39	-8.39	0	8.31	-8.31
35	10.7	15.06	-4.36	19.14	17.47	1.67	13.31	14.76	-1.45
36	0.06	-0.12	0.18	0.55	13.44	-12.89	0.03	9.46	-9.43
37	6.86	25.76	-18.9	20.25	19.58	0.67	14.39	16.59	-2.2
38	0	0.39	-0.39	7.06	13.06	-6	0.21	10.28	-10.07
39	0	3.21	-3.21	15.95	11.58	4.37	24.05	26.53	-2.48
40	7.14	13.84	-6.7	14.4	16.73	-2.33	10.04	13.88	-3.84
41	2.94	6.98	-4.04	1.6	14.75	-13.15	3.97	9.18	-5.21
42	8.44	10.76	-2.32	13.65	24.7	-11.05	9.47	16.48	-7.01
43	14.16	8.07	6.09	14.55	21.32	-6.77	33.66	14.76	18.9
44	5.66	10.34	-4.68	8.73	9.36	-0.63	20.04	15.24	4.8
45	2.44	4.52	-2.08	23.69	19.93	3.76	23.93	19.53	4.4
46	163.53	67.26	96.27	0	8.01	-8.01	0	18.47	-18.47
47	0	2.22	-2.22	21.85	18.36	3.49	18.16	17.08	1.08
48	0	13	-13	0	20.27	-20.27	50	28.23	21.77
49	22.91	20.9	2.01	60	30.54	29.46	0	26.95	-26.95
50	0	21.23	-21.23	39.88	29.84	10.04	0.12	24.17	-24.05
51	3.73	0.58	3.15	0	14.17	-14.17	40	13.8	26.2
52	0	8.68	-8.68	0	24.28	-24.28	50	21.31	28.69
53	0	1.82	-1.82	29.93	22.64	7.29	0.07	14.42	-14.35
54	80.95	46.66	34.29	0	10.36	-10.36	0	26.38	-26.38

Appendix C: True values and predicted values comparison - test dataset – single-phase

	Gas_true	Gas_preds	Gas error	Oil true	Oil_preds	Oil error	Water_true	Water_preds	Water error
0	0	0.76	-0.76	0	3.2	-3.2	20	13.84	6.16
1	0	2.98	-2.98	0	13.86	-13.86	9.91	9.33	0.57
2	0	0.74	-0.74	0	2.43	-2.43	39.99	24.34	15.65
3	80.95	0.8	80.15	0	5.76	-5.76	0	31.17	-31.17
4	0	11.39	-11.39	7.06	8.18	-1.12	0.21	12.92	-12.71
5	0	0.79	-0.79	0	4.33	-4.33	30	17.66	12.34

Appendix D: True values and predicted values comparison - test dataset – Multiphase model (three accelerometer)

	Gas_true	Gas_preds	Gas Error	Oil_true	Oil_preds	Oil Error	Water_true	Water_preds	Water Error
0	5.45	7.73	-2.28	7.8	7.54	0.26	17.81	16.4	1.41
1	18.49	19.41	-0.92	52.93	19.23	33.7	35.32	30.8	4.52
2	26.62	18.54	8.08	52.8	26.52	26.28	35.2	18.16	17.04
3	3.68	3.69	-0.01	13.63	8.82	4.81	14.16	9.91	4.25
4	6.71	14.86	-8.15	12.16	10.48	1.68	12.53	17.26	-4.73
5	11.13	9.62	1.51	19.92	17.86	2.06	20.09	10.97	9.12
6	14.09	15.84	-1.75	15.58	18.92	-3.34	10.89	9.58	1.31
7	0.16	-2.36	2.52	10.57	7.99	2.58	1.5	4.9	-3.4
8	4.07	-1.59	5.66	34.8	11.86	22.94	14.95	16.29	-1.34
9	3.29	7.55	-4.26	14.25	28.21	-13.96	33.35	27.51	5.84
10	5.64	0.74	4.9	13.61	9.72	3.89	31.56	16.29	15.27
11	0	-0.5	0.5	25.91	16.56	9.35	14.1	14.5	-0.4
12	0	9.68	-9.68	0	15.88	-15.88	50	35.49	14.51
13	0	11.38	-11.38	32.91	30.2	2.71	27.33	37.97	-10.64
14	13.4	10.72	2.68	23.8	16.46	7.34	24.18	15.89	8.29
15	20.27	12.58	7.69	34.7	16.75	17.95	23.31	39.84	-16.53
16	10.34	14.25	-3.91	6	17.2	-11.2	13.71	6.72	6.99
17	14.58	13.47	1.11	10.37	16.81	-6.44	24.93	15.39	9.54
18	32.37	23.47	8.9	15.57	17.15	-1.58	42.52	25.61	16.91
19	7.41	14.95	-7.54	0	13.99	-13.99	40	14.42	25.58
20	9.93	12.58	-2.65	0	20.81	-20.81	43.34	20.17	23.17
21	14.24	12.89	1.35	16.92	42.38	-25.46	40.19	24.24	15.95
22	0	6.1	-6.1	25.92	5.73	20.19	14.06	16.55	-2.49
23	0.18	0.64	-0.46	15.05	12.82	2.23	6.42	19.5	-13.08
24	0	3.97	-3.97	0	10.7	-10.7	20	11.8	8.2
25	4.92	12.21	-7.29	2.6	11.24	-8.64	6.37	0.83	5.54
26	10.22	1.57	8.65	11.83	17.47	-5.64	8.18	11.14	-2.96
27	26.34	24.89	1.45	28.8	27.08	1.72	31.19	18.43	12.76
28	11.6	9.04	2.56	28.07	12.86	15.21	20.77	18.42	2.35
29	19.75	6.49	13.26	0	16.52	-16.52	0	20.29	-20.29
30	9.43	12.16	-2.73	4.9	7.81	-2.91	12.01	5.4	6.61
31	0	-1.58	1.58	11.91	22.35	-10.44	28.09	13	15.09
32	9.9	10.28	-0.38	16.84	22.85	-6.01	40.27	20.13	20.14
33	0.05	1.64	-1.59	16.98	26.11	-9.13	40.12	21.58	18.54
34	0.06	2.54	-2.48	0	8.21	-8.21	0	4.55	-4.55
35	10.7	17.66	-6.96	19.14	28.86	-9.72	13.31	16.97	-3.66
36	0.06	-2.61	2.67	0.55	14.88	-14.33	0.03	5.63	-5.6
37	6.86	12.87	-6.01	20.25	12.44	7.81	14.39	10.78	3.61

38	0	-0.04	0.04	7.06	12.71	-5.65	0.21	11.02	-10.81
39	0	1.73	-1.73	15.95	11.51	4.44	24.05	25.71	-1.66
40	7.14	11.56	-4.42	14.4	11.97	2.43	10.04	8.84	1.2
41	2.94	3.54	-0.6	1.6	5.4	-3.8	3.97	3.84	0.13
42	8.44	12.83	-4.39	13.65	18.44	-4.79	9.47	7.17	2.3
43	14.16	14.24	-0.08	14.55	18.14	-3.59	33.66	12.37	21.29
44	5.66	9.8	-4.14	8.73	12.41	-3.68	20.04	10.98	9.06
45	2.44	1.19	1.25	23.69	25.07	-1.38	23.93	14.96	8.97
46	163.53	45.91	117.62	0	5.41	-5.41	0	39.48	-39.48
47	0	-1.58	1.58	21.85	14.15	7.7	18.16	17.16	1
48	0	12.32	-12.32	0	15.87	-15.87	50	34.52	15.48
49	22.91	22.64	0.27	60	14.18	45.82	0	23.83	-23.83
50	0	17.91	-17.91	39.88	25.58	14.3	0.12	26.92	-26.8
51	3.73	1.44	2.29	0	21.4	-21.4	40	19.24	20.76
52	0	5.19	-5.19	0	14.56	-14.56	50	21.59	28.41
53	0	-0.32	0.32	29.93	16.04	13.89	0.07	13.57	-13.5
54	80.95	19.01	61.94	0	15.66	-15.66	0	18.89	-18.89

Appendix E: True values and predicted values comparison - test dataset – Single-phase model (three accelerometer)

	Gas_true	Gas_preds	Gas Error	Oil_true	Oil_preds	Oil Error	Water_true	Water_preds	Water Error
0	0	0	0	0	5.65	-5.65	20	9.41	10.59
1	0	6.67	-6.67	0	13.86	-13.86	9.91	4.55	5.36
2	0	-0.02	0.02	0	5.74	-5.74	39.99	20.78	19.21
3	80.95	-0.04	80.99	0	6.57	-6.57	0	25.17	-25.17
4	0	9.45	-9.45	7.06	5.2	1.86	0.21	22.91	-22.7
5	0	0.22	-0.22	0	7.09	-7.09	30	13.22	16.78

Appendix F: True values and predicted values comparison - test dataset – Physics-based multiphase model

	Gas true	Gas preds	Gas error	Oil true	Oil preds	Oil error	Water true	Water preds	Water error
0	5.45	2.89	2.56	7.8	13.07	-5.27	17.81	12.36	5.45
1	18.49	33.6	-15.11	52.93	17.49	35.44	35.32	34.98	0.34
2	26.62	25.02	1.6	52.8	26.09	26.71	35.2	33.35	1.85
3	3.68	7.46	-3.78	13.63	19.93	-6.3	14.16	5.97	8.19
4	6.71	15.3	-8.59	12.16	13.46	-1.31	12.53	14.78	-2.25
5	11.13	11.74	-0.6	19.92	17.2	2.72	20.09	5.81	14.28
6	14.09	19.18	-5.09	15.58	29.32	-13.74	10.89	19.12	-8.22
7	0.16	3.82	-3.66	10.57	9.44	1.14	1.5	9.62	-8.12
8	4.07	2.17	1.9	34.8	12.98	21.82	14.95	15.74	-0.79
9	3.29	11.61	-8.32	14.25	20.96	-6.71	33.35	24.7	8.64
10	5.64	8.78	-3.14	13.61	20.17	-6.56	31.56	30.41	1.14
11	0	6.87	-6.87	25.91	16.27	9.64	14.1	9.83	4.27
12	0	8.65	-8.65	0	20.8	-20.8	50	44.17	5.84
13	0	8.69	-8.69	32.91	10.55	22.36	27.33	26.34	0.98
14	13.4	7.07	6.33	23.8	21.28	2.52	24.18	10.04	14.14
15	20.27	22.76	-2.49	34.7	15.34	19.36	23.31	37.09	-13.78
16	10.34	4.41	5.93	6	15.88	-9.88	13.71	15.24	-1.53
17	14.58	11.29	3.29	10.37	14.4	-4.02	24.93	17.2	7.73
18	32.37	21.61	10.76	15.57	27.16	-11.58	42.52	24.75	17.77
19	7.41	8.86	-1.45	0	18.07	-18.07	40	10.86	29.14
20	9.93	10.79	-0.86	0	20.68	-20.68	43.34	14.07	29.27
21	14.24	15.63	-1.39	16.92	33.59	-16.68	40.19	24.37	15.82
22	0	8.91	-8.91	25.92	15.06	10.86	14.06	12.84	1.22
23	0.18	5.22	-5.05	15.05	14.62	0.43	6.42	16.77	-10.34
24	0	6.71	-6.71	0	16.04	-16.04	20	19.34	0.66
25	4.92	12.61	-7.69	2.6	14.66	-12.06	6.37	11.91	-5.54
26	10.22	6.94	3.28	11.83	15.42	-3.58	8.18	16.56	-8.38
27	26.34	9.11	17.22	28.8	27.8	1	31.19	9.96	21.24
28	11.6	2.45	9.15	28.07	17.08	10.99	20.77	21.53	-0.76
29	19.75	28.37	-8.63	0	15.95	-15.95	0	34.27	-34.27
30	9.43	6.33	3.1	4.9	7.61	-2.71	12.01	6.57	5.44
31	0	4.48	-4.48	11.91	16.66	-4.74	28.09	9.36	18.72
32	9.9	9.62	0.28	16.84	25.81	-8.98	40.27	21.47	18.79
33	0.05	4.35	-4.3	16.98	15.99	0.99	40.12	18.87	21.25
34	0.06	5.9	-5.84	0	10.36	-10.36	0	7.83	-7.83
35	10.7	19.04	-8.34	19.14	18.42	0.71	13.31	20.02	-6.71
36	0.06	4.07	-4.01	0.55	6.73	-6.17	0.03	10.64	-10.62
37	6.86	14.53	-7.67	20.25	12.89	7.36	14.39	16.49	-2.1

38	0	4.22	-4.22	7.06	16.25	-9.19	0.21	9.36	-9.15
39	0	6.05	-6.05	15.95	8.83	7.12	24.05	21.5	2.54
40	7.14	13.28	-6.13	14.4	19.21	-4.81	10.04	13.29	-3.25
41	2.94	10.64	-7.7	1.6	13.73	-12.12	3.97	11.69	-7.72
42	8.44	7.6	0.84	13.65	24.25	-10.6	9.47	14.93	-5.45
43	14.16	19.46	-5.3	14.55	26.33	-11.78	33.66	20.25	13.4
44	5.66	15.62	-9.96	8.73	9.96	-1.23	20.04	14.05	6
45	2.44	7.9	-5.46	23.69	17.87	5.82	23.93	23.9	0.03
46	163.53	59.16	104.36	0	15.99	-15.99	0	17.15	-17.15
47	0	11.34	-11.34	21.85	9.26	12.59	18.16	16.98	1.18
48	0	13.25	-13.25	0	21.75	-21.75	50	40.46	9.54
49	22.91	15.37	7.54	60	28.39	31.61	0	21.5	-21.5
50	0	21.36	-21.36	39.88	30.86	9.02	0.12	28.89	-28.77
51	3.73	8.18	-4.44	0	14.36	-14.36	40	18.46	21.54
52	0	12.55	-12.55	0	23.27	-23.27	50	27.71	22.29
53	0	1.41	-1.41	29.93	24.44	5.48	0.07	12.69	-12.61
54	80.95	39.74	41.21	0	19.28	-19.28	0	28.55	-28.55

Appendix G: Code for feature extraction

```
# Load the data
data = np.loadtxt(file_path)

# Use Short-Time Fourier Transform (STFT) to extract the frequency components
f, t, Zxx = signal.stft(data, fs=1/0.000019531, nperseg=50000*5)

# Identify the frequency bins that contain the noise and its harmonics
noise_bins = np.where((f > 190) & (f < 210))[0]
harmonics_bins = np.where((f > 380) & (f < 420))[0]
harmonics_bins_2 = np.where((f > 760) & (f < 840))[0]
combined_bins = np.concatenate((noise_bins, harmonics_bins, harmonics_bins_2))

# Apply a binary mask to the STFT output to zero out the noise and its harmonics
mask = np.ones(Zxx.shape)
mask[combined_bins] = 0

# Inverse STFT to obtain the denoised signal
_, denoised_data = signal.istft(mask * Zxx)

# Convert the denoised data back to the original time domain
denoised_data = np.real(denoised_data)

# performing fft on filtered data/ denoised data
sr = 1/0.000019531 # sampling rate (Hz)
X = scipy.fft.fft(denoised_data) # fft performed
n = np.arange(len(X)) #length of fft
T = len(X)/sr # sampling period
x_freq = n/T

# find peaks
peaks, props = scipy.signal.find_peaks(np.abs(X[:int(len(X)*20000//sr)]), height=5, distance=100000)

# Get the frequencies of the peaks
peak_amplitudes = props['peak_heights']
peak_frequencies = x_freq[peaks]
# Sort the peak amplitudes in descending order
sorted_indices = np.argsort(-peak_amplitudes)

# Get the top 17 frequencies
top_17_frequencies = peak_frequencies[sorted_indices[:17]]
```

Appendix H: Code for multiphase flow Neural Network

```
def build_model():
    # Define model layers.
    input_layer = Input(shape=(len(train.columns),))
    first_dense = Dense(units='128', activation='relu')(input_layer)

    second_dense_1 = Dense(units='128', activation='relu')(first_dense)
    third_dense_1 = Dense(units='128', activation='relu')(second_dense_1)
    fourth_dense_1 = Dense(units='128', activation='relu')(third_dense_1)
    fifth_dense_1 = Dense(units='128', activation='relu')(fourth_dense_1)
    # Y1 output will be fed from the fifth dense 1
    y1_output = Dense(units='1', name='gas_output')(fifth_dense_1)

    second_dense_2 = Dense(units='128', activation='relu')(first_dense)
    third_dense_2 = Dense(units='128', activation='relu')(second_dense_2)
    fourth_dense_2 = Dense(units='128', activation='relu')(third_dense_2)
    fifth_dense_2 = Dense(units='128', activation='relu')(fourth_dense_2)
    sixth_dense_2 = Dense(units='128', activation='relu')(fifth_dense_2)
    # Y2 output will be fed from the sixth dense 2
    y2_output = Dense(units='1', name='oil_output')(sixth_dense_2)

    second_dense_3 = Dense(units='128', activation='relu')(first_dense)
    third_dense_3 = Dense(units='128', activation='relu')(second_dense_3)
    fourth_dense_3 = Dense(units='128', activation='relu')(third_dense_3)
    # Y3 output will be fed from the fourth dense 3
    y3_output = Dense(units='1', name='water_output')(fourth_dense_3)

    # Define the model with the input layer and a list of output layers
    model = Model(inputs=input_layer, outputs=[y1_output, y2_output,   y3_output])

    return model

model = build_model()
# Specifying the optimizer, and compiling the model with loss functions for all 3 outputs
optimizer = tf.keras.optimizers.Adam(learning_rate=0.00015)
model.compile(optimizer=optimizer,
          loss={'gas_output': 'mse', 'oil_output': 'mse', 'water_output': 'mse'},
          metrics={'gas_output': tf.keras.metrics.RootMeanSquaredError(),
                'oil_output': tf.keras.metrics.RootMeanSquaredError(),
                'water_output': tf.keras.metrics.RootMeanSquaredError()})

# Train the model
history = model.fit(norm_train_X, train_Y,
              epochs=8,
              batch_size=2,
              validation_data=(norm_val_X, val_Y))
```

Appendix I: Code for single-phase flow prediction neural network

```
def build_model():
    # Define model layers.
    input_layer = Input(shape=(len(train.columns),))
    first_dense = Dense(units='128', activation='relu')(input_layer)

    second_dense_1 = Dense(units='128', activation='relu')(first_dense)
    third_dense_1 = Dense(units='128', activation='relu')(second_dense_1)
    fourth_dense_1 = Dense(units='128', activation='relu')(third_dense_1)
    fifth_dense_1 = Dense(units='128', activation='relu')(fourth_dense_1)
    sixth_dense_1 = Dense(units='128', activation='relu')(fifth_dense_1)
    seventh_dense_1 = Dense(units='128', activation='relu')(sixth_dense_1)
    eight_dense_1 = Dense(units='128', activation='relu')(seventh_dense_1)
    ninth_dense_1 = Dense(units='128', activation='relu')(eight_dense_1)
    # Y1 output will be fed from the ninth dense 1
    y1_output = Dense(units='1', name='gas_output')(ninth_dense_1)

    second_dense_2 = Dense(units='128', activation='relu')(first_dense)
    third_dense_2 = Dense(units='128', activation='relu')(second_dense_2)
    fourth_dense_2 = Dense(units='128', activation='relu')(third_dense_2)
    fifth_dense_2 = Dense(units='128', activation='relu')(fourth_dense_2)
    # Y2 output will be fed from the fifth dense 2
    y2_output = Dense(units='1', name='oil_output')(fifth_dense_2)

    second_dense_3 = Dense(units='128', activation='relu')(first_dense)
    third_dense_3 = Dense(units='128', activation='relu')(second_dense_3)
    # Y3 output will be fed from the fourth dense 3
    y3_output = Dense(units='1', name='water_output')(third_dense_3)

    # Define the model with the input layer and a list of output layers
    model = Model(inputs=input_layer, outputs=[y1_output, y2_output, y3_output])

    return model
```

Appendix J: Code for multiphase flow prediction neural network (trained with three accelerometer data)

```
def build_model():
    # Define model layers.
    input_layer = Input(shape=(len(train.columns),))
    first_dense = Dense(units='128', activation='relu')(input_layer)

    second_dense_1 = Dense(units='128', activation='relu')(first_dense)
    third_dense_1 = Dense(units='128', activation='relu')(second_dense_1)
    fourth_dense_1 = Dense(units='128', activation='relu')(third_dense_1)
    fifth_dense_1 = Dense(units='128', activation='relu')(fourth_dense_1)
    sixth_dense_1 = Dense(units='128', activation='relu')(fifth_dense_1)
    # Y1 output will be fed from the sixth dense 1
    y1_output = Dense(units='1', name='gas_output')(sixth_dense_1)

    second_dense_2 = Dense(units='128', activation='relu')(first_dense)
    third_dense_2 = Dense(units='128', activation='relu')(second_dense_2)
    fourth_dense_2 = Dense(units='128', activation='relu')(third_dense_2)
    fifth_dense_2 = Dense(units='128', activation='relu')(fourth_dense_2)
    sixth_dense_2 = Dense(units='128', activation='relu')(fifth_dense_2)
    # Y2 output will be fed from the sixth dense 2
    y2_output = Dense(units='1', name='oil_output')(sixth_dense_2)

    second_dense_3 = Dense(units='128', activation='relu')(first_dense)
    third_dense_3 = Dense(units='128', activation='relu')(second_dense_3)
    fourth_dense_3 = Dense(units='128', activation='relu')(third_dense_3)
    # Y3 output will be fed from the fourth dense 3
    y3_output = Dense(units='1', name='water_output')(fourth_dense_3)

    # Define the model with the input layer and a list of output layers
    model = Model(inputs=input_layer, outputs=[y1_output, y2_output, y3_output])

    return model
```

Appendix K: Code for single-phase flow rate prediction neural network (trained with three accelerometer data)

```
def build_model():
    # Define model layers.
    input_layer = Input(shape=(len(train.columns),))
    first_dense = Dense(units='128', activation='relu')(input_layer)

    second_dense_1 = Dense(units='128', activation='relu')(first_dense)
    third_dense_1 = Dense(units='128', activation='relu')(second_dense_1)
    fourth_dense_1 = Dense(units='128', activation='relu')(third_dense_1)
    fifth_dense_1 = Dense(units='128', activation='relu')(fourth_dense_1)
    sixth_dense_1 = Dense(units='128', activation='relu')(fifth_dense_1)
    seventh_dense_1 = Dense(units='128', activation='relu')(sixth_dense_1)
    eight_dense_1 = Dense(units='128', activation='relu')(seventh_dense_1)
    ninth_dense_1 = Dense(units='128', activation='relu')(eight_dense_1)
    # Y1 output will be fed from the ninth dense 1
    y1_output = Dense(units='1', name='gas_output')(ninth_dense_1)

    second_dense_2 = Dense(units='128', activation='relu')(first_dense)
    third_dense_2 = Dense(units='128', activation='relu')(second_dense_2)
    fourth_dense_2 = Dense(units='128', activation='relu')(third_dense_2)
    fifth_dense_2 = Dense(units='128', activation='relu')(fourth_dense_2)
    # Y2 output will be fed from the fifth dense 2
    y2_output = Dense(units='1', name='oil_output')(fifth_dense_2)

    second_dense_3 = Dense(units='128', activation='relu')(first_dense)
    third_dense_3 = Dense(units='128', activation='relu')(second_dense_3)
    # Y3 output will be fed from the third dense 3
    y3_output = Dense(units='1', name='water_output')(third_dense_3)

    # Define the model with the input layer and a list of output layers
    model = Model(inputs=input_layer, outputs=[y1_output, y2_output, y3_output])

    return model
```

Appendix L: Code for spectrogram generation automated script

```
import os
import pandas as pd
from scipy.signal import spectrogram
import matplotlib.pyplot as plt
import numpy as np
import time
# Set the directory containing the data files
data_dir = 'RAW_DATA/GO'

# Set the directory to save spectrograms
spec_dir = 'SPECTROGRAMS_15/GO'

# Set the sampling rate
fs = 1/0.000019531 # sampling rate (Hz)

# Loop through each file in the data directoryW
for file in os.listdir(data_dir):
    plt.clf()
    # Check if file is a CSV file
    if file.endswith('.csv'):
        # Read the CSV file
        df = pd.read_csv(os.path.join(data_dir, file), header=None)


        # Convert the data to a NumPy array, use only 10 second data
        data = df.values[:int(10 * fs)].flatten()


        # Compute the spectrogram
        f, t, Sxx = spectrogram(data, fs)


        # Plot and save the spectrogram
        plt.pcolormesh(t, f, np.log10(Sxx), vmin=-9.9, vmax=-5, shading='gouraud')
        plt.ylim(0, 20000)  # set the y-axis limit
        # Remove tick labels and ticks
        plt.xticks([])
        plt.yticks([])
        plt.tick_params(axis='both', which='both', length=0)
        # Save image without paddings
        plt.savefig(os.path.join(spec_dir, os.path.splitext(file)[0] + '.png'), bbox_inches='tight', pad_inches=0)
        plt.close()
        # Delete variables to free up memory
        del df, data, f, t, Sxx
        time.sleep(0.5)
```

Appendix M: Code for splitting spectrogram dataset into train and test set

```
import splitfolders

input_folder = "SPECTROGRAMS_15" #Enter Input Folder
output = "images_recognition/train_test_17" #Enter Output Folder

splitfolders.ratio(input_folder, output=output, seed=42, ratio=(0.85, 0.15))
```

Appendix N: Code for CNN image recognition model

```
from tensorflow.keras import layers
from tensorflow.keras.layers.experimental import preprocessing
from tensorflow.keras.models import Sequential

# Setup data augmentation
data_augmentation = Sequential([
   preprocessing.RandomFlip("horizontal")
], name ="data_augmentation")

# Setup the base model
base_model = tf.keras.applications.EfficientNetB2(include_top=False)
base_model.trainable = False

# Setup model architecture with trainable top layers
inputs = layers.Input(shape=(224, 224, 3), name="input_layer")
x = data_augmentation(inputs)
x = base_model(x, training=False)
x = layers.GlobalAveragePooling2D(name="global_avg_pool_layer")(x)
outputs = layers.Dense(7, activation="softmax", name="output_layer")(x)
model = tf.keras.Model(inputs, outputs)

# Compile
model.compile(loss="categorical_crossentropy",
          optimizer=tf.optimizers.Adam(),
          metrics=["accuracy"])

# Fit
history_model = model.fit(train_data,
                      epochs=14,
                      validation_data=test_data,
                      validation_steps=len(test_data))
```

Appendix O: Code for physics-based multiphase flow rate prediction model with customized loss function

```
L = 4 # distance between accelerometers in the rig in meters
d = 0.0762 # inner diameter of rig in meters
g = 9.8 # gravitional acceleration in m^-2
# Define loss function using the Darcy-Weisbach equation
def custom_loss(y_true, y_pred):
    # Calculate the head loss using the Darcy-Weisbach equation
    v = tf.abs(y_true - y_pred)
    head_loss = ((L/d) * (v**2 / (2*g)))
    # Check for NaN values and set them to 0
    head_loss = tf.where(tf.math.is_nan(head_loss), tf.zeros_like(head_loss), head_loss)
    # Return the mean squared error of the head loss
    return tf.keras.backend.mean(tf.keras.backend.square(head_loss))

def build_model():
    # Define model layers.
    input_layer = Input(shape=(len(train.columns),))
    first_dense = Dense(units='128', activation='relu')(input_layer)

    second_dense_1 = Dense(units='128', activation='relu')(first_dense)
    third_dense_1 = Dense(units='128', activation='relu')(second_dense_1)
    fourth_dense_1 = Dense(units='128', activation='relu')(third_dense_1)
    fifth_dense_1 = Dense(units='128', activation='relu')(fourth_dense_1)
    # Y1 output will be fed from the fifth dense 1
    y1_output = Dense(units='1', name='gas_output')(fifth_dense_1)

    second_dense_2 = Dense(units='128', activation='relu')(first_dense)
    third_dense_2 = Dense(units='128', activation='relu')(second_dense_2)
    fourth_dense_2 = Dense(units='128', activation='relu')(third_dense_2)
    # Y2 output will be fed from the fourth dense 2
    y2_output = Dense(units='1', name='oil_output')(fourth_dense_2)

    second_dense_3 = Dense(units='128', activation='relu')(first_dense)
    third_dense_3 = Dense(units='128', activation='relu')(second_dense_3)
    fourth_dense_3 = Dense(units='128', activation='relu')(third_dense_3)
    # Y3 output will be fed from the fourth dense 3
    y3_output = Dense(units='1', name='water_output')(fourth_dense_3)

    # Define the model with the input layer and a list of output layers
    model = Model(inputs=input_layer, outputs=[y1_output, y2_output, y3_output])
    return model

model = build_model()
# Specifying the optimizer, and compiling the model with custom loss functions for all 3 outputs
optimizer = tf.keras.optimizers.Adam(learning_rate=0.00015)
model.compile(optimizer=optimizer,
              loss={'gas_output': custom_loss, 'oil_output': custom_loss, 'water_output': custom_loss},
              metrics={'gas_output': tf.keras.metrics.RootMeanSquaredError(),
                       'oil_output': tf.keras.metrics.RootMeanSquaredError(),
                       'water_output': tf.keras.metrics.RootMeanSquaredError()})
```