Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

Christian Danh Nguyen

Deep Learning Based People Estimation on 2D Ultra-Wideband Radar Data

Master's thesis in Electronic Systems Design
Supervisor: Pierluigi Salvo Rossi
July 2023

NTNU
Norwegian University of Science and Technology

NOVELDA

Christian Danh Nguyen

Deep Learning Based People Estimation on 2D Ultra-Wideband Radar Data

Master's thesis in Electronic Systems Design
Supervisor: Pierluigi Salvo Rossi
July 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU
Norwegian University of
Science and Technology

Abstract

This master's thesis investigates the performance of three types of deep learning models, the Simple Convolutional Neural Network (CNN), the Residual Network (ResNet), and the Convolutional Neural Network with Gated Recurrent Units (CNN+GRU), in the task of detecting and counting people using range-Doppler (RD) maps obtained from the novel NOVELDA Ultra-low Power Presence Sensor. The dual-antenna of the sensor allowed for digital beamforming, creating three beams pointing in $\pm 20^\circ$ and 0° azimuth angle. The results suggest that the CNN model, with its aptitude for spatial feature extraction, performed well on the RD maps, which spatially represent radar reflections from people in motion. This characteristic may not have been fully exploited by the CNN+GRU model, despite its potential to capture temporal dependencies, or by the ResNet model, which might have been too intricate for this dataset.

A significant limitation of this study was the inadequate training data for the multiclass classification of RD maps for people counting. Insufficient data might have led to overfitting, negatively impacting the model's ability to generalize, which was evident in the performances of the more complex models, ResNet and CNN+GRU. To address this, data augmentation techniques were used to enhance the diversity of the training set, although their effectiveness might be limited.

The height of the radar system, fixed at 1.5 m in the experiments, could significantly influence the characteristics of the radar return signals, particularly under Non-Line-Of-Sight (NLOS) conditions. Exploring the effects of radar height placement could be a future research direction. Furthermore, minimal preprocessing to remove noise and clutter from the RD maps was conducted. In some cases, the presence of noise and clutter could potentially carry useful information that aids the model in distinguishing between noise and target signals.

Despite the above challenges, the CNN model yielded promising results for the classification of up to three targets in a 4×4 m grid, that are comparable to state-of-the-art methods. Utilizing the NOVELDA Ultra-low Power Presence Sensor demonstrates the potential to implement radar systems with lower power consumption and improved target localization and differentiation capabilities, thanks to its dual-antenna configuration. Beamforming also provides insights into target dynamics and helps to resolve issues of target occlusions and overlapping trajectories, offering more accurate and reliable people counting results.

Sammendrag

Denne masteroppgaven undersøker ytelsen til tre typer maskinlæringsmodeller, Convolutional Neural Network (CNN), Residual Network (ResNet), og Convolutional Neural Network med Gated Recurrent Units (CNN+GRU), i oppgaven med å detektere og telle mennesker ved bruk av range-Doppler (RD) kart hentet fra den nye NOVELDA Ultra-low Power Presence Sensoren. Sensorens doble antenne tillot digital stråledannelse, som skaper tre stråler som peker i -20°, 0°, og +20° asimutvinkel. Resultatene antyder at CNN-modellen, med sin dyktighet for romlige egenskaper, presterte godt på RD-kartene, som representerer radarrefleksjoner fra mennesker i bevegelse. Denne egenskapen kan ikke ha blitt fullt utnyttet av CNN+GRU-modellen, til tross for dens potensiale for å fange opp avhengigheter i tid, eller av ResNet-modellen, som kanskje har vært for intrikat for dette datasettet.

En betydelig begrensning i denne studien var det utilstrekkelige treningsdatasettet for flerklassifisering av RD-kart for mennesketelling. Utilstrekkelige data kan ha ført til overtilpasning, noe som negativt påvirker modellens evne til å generalisere, noe som var tydelig i ytelsen til de mer komplekse modellene, ResNet og CNN+GRU. For å løse dette, ble teknikker for dataforsterkning brukt for å øke mangfoldet i treningssettet, selv om deres effektivitet kan være begrenset.

Høyden på radarsystemet, fastsatt til 1,5 m i eksperimentene, kan påvirke egenskapene til radarretursignalene betydelig, spesielt under ikke-line-of-sight (NLOS) forhold. Å utforske effektene av radarhøydeplassering kan være en fremtidig forskningsretning. Videre ble det utført minimal forbehandling for å fjerne støy og clutter fra RD-kartene. I noen tilfeller kan tilstedeværelsen av støy og clutter potensielt bære nyttig informasjon som hjelper modellen i å skille mellom støy og målsignaler.

Til tross for de ovennevnte utfordringene, ga CNN-modellen lovende resultater for klassifiseringen av opptil tre mål i et 4x4 m rutenett, som er sammenlignbare med toppmoderne metoder. Bruken av NOVELDA Ultra-low Power Presence Sensoren viser potensialet til å implementere radarsystemer med lavere strømforbruk og forbedrede mållokalisering og differensieringsegenskaper, takket være dens doble antennekonfigurasjon. Stråledannelse gir også innsikt i måldynamikk og bidrar til å løse problemer med målokklusjoner og overlappende baner, noe som gir mer nøyaktige og pålitelige resultater for mennesketelling.

Preface

I would like to dedicate my thesis to my friends, girlfriend and family, who have supported me throughout my studies. I would also like to thank my supervisor at NTNU, Professor Pierluigi Salvo Rossi, and my external supervisor at Novelda, Jan Roar Pleym, for their guidance and support throughout the project. Furthermore I would like thank Sigurd Pleym, Benjamin Nedregård, Shaurya Sharma, and the rest of the Novelda team for their help with the data collection and annotation work.

Today is the 21st of July, 2023. I am currently sitting in a hotel in Saigon, Vietnam, five days into the vacation with my high school friends, Felix Gia-Bao Hoang, Varshan Erik Shankar, and Emil Gravningen Pilley. The vacation was planned together in November as a celebration of mine and Varshan's completion (hopefully) of a M.Sc. degree at NTNU, but I haven't even gotten the slightest tan and am mistaken for either a Korean, Chinese or Japanese here. I am Vietnamese.

In Türkiye, during our transit, a receptionist, who was incredibly well spoken in various languages, tried to catch my attention by greeting me: "Anyeong haseyo" (Korean). I told her calmly that I was not korean, and told her that we say "hei" to greet people where I am from (Norway). In response she said: "Hai, hajimemashite".

I digress. I have been sitting inside the hotel room for days, finishing up this master's thesis, and today I am finally done. It has been a ride, and I am looking forward to start my last vacation.

-Christian Danh Nguyen

Table of contents

List of figures

List of tables

List of abbreviations and symbols

Abbreviations

1D	One dimensional
2D	Two dimensional
3D	Three dimensional
CFAR	Constant false alarm rate
CNN	Convolutional neural network
CPI	Coherent processing interval
DFT	Discrete fourier transform
DSP	Digital signal processing
FFT	Fast fourier transform
GAN	Generative adversarial network
IIR	Infinite impulse response
IR	Impulse radio
ML	Machine learning
PRF	Pulse repetition frequency
RAI	Range-angle image
RDI	Range-doppler image
Rx	Receiver
Tx	Transmitter
UWB	Ultra-wideband
VAE	Variational autoencoder

Symbols

θ_3	3-dB beamwidth

c_0	Speed of light
f_D	Doppler frequency
R_max	Maximum unambiguous range
v_r	Unambiguous radial velocity
$\Delta\theta$	Azimuth angle resolution
ΔR	Range resolution
λ	Wavelength
ϕ	Elevation angle
θ	Azimuth angle
B	Bandwidth [Hz]

1 Introduction

1.1 Background and motivation

In recent years, the emergence of Impulse Radio Ultra-Wideband (IR-UWB) technology has become more apparent in various fields, including healthcare, robotics, and automotive applications [1]–[5]. Its ability to offer high-precision, anonymous and non-contact sensing capabilities makes it an attractive solution for a wide range of real-world problems. An example of this technology for medical applications is UWB medical imaging, which benefits in providing low-risk imaging of the internal organs and tissues of the human body [4]. UWB radars can also be used for non-contact breathing monitoring [5], and to detect sleep stages [6].

One particularly promising application is people counting (PC), which can have significant implications in various domains such as smart buildings by saving energy based on inefficient lighting and heating of unoccupied spaces, privacy by not being intrusive compared to cameras, and crowd management by redirecting crowds in public areas and transport. PC using IR-UWB radar have given promising results [7]–[10]. However, they are still not robust enough to be reliably deployed in the real world.

This master's thesis explores the use of machine learning (ML) techniques on range-Doppler (RD) maps obtained from IR-UWB radar for accurate PC in order to cover the limitations of traditional approaches.

At the core of this study lies NOVELDA's new Ultra-low Power Presence Sensor, a cutting-edge IR-UWB radar sensor designed for low-power operation of below 100 microwatts and accurate sensing. The sensor offers a dual-antenna radar system, which allows for two-dimensional spatial information: range and direction-of-arrival in azimuth angle. Leveraging this state-of-the-art radar sensor, we aim to develop and evaluate a robust and efficient ML model for precise PC.

To achieve accurate PC results, extensive data preprocessing is essential. As part of this study, we will investigate various preprocessing steps for the RD maps, with a particular focus on minimal noise filtering. By allowing the model to learn the noise and clutter floor inherent in real-world radar data, we seek to improve the robustness of our ML approach.

Data collection plays a pivotal role in training and evaluating ML models. In this study, we will also conduct our data collection using NOVELDA's Ultra-low Power Presence Sensor and design specific scenarios to generate diverse RD maps reflecting real-world and challenging conditions. Ground truth labels for the number of people in each scenario will be collected, and we will present the methods employed for accurate annotation.

The research questions addressed in the thesis are as follows:

- IR-UWB technology be utilized for people counting, and what are its unique advantages and limitations in this context?
- What are the optimal preprocessing steps for RD maps derived from IR-UWB radar data, particularly in terms of minimal noise filtering and clutter floor management, to improve the accuracy and generalization of the machine learning model?
- How can the Ultra-low Power Presence Sensor by NOVELDA be effectively integrated with machine learning techniques to create a reliable PC model, and how does its performance compare to traditional approaches?
- What data collection methods can be employed to generate diverse and representative RD maps, and what are the implications of data quality and quantity on the performance of the machine learning model?

Through this thesis, we envision contributing valuable insights into the application of IR-UWB radar technology and its seamless integration with ML for PC. The combination of cutting-edge

radar sensors, data preprocessing techniques, and novel data collection methods will be instrumental in developing an efficient and reliable solution for PC in a variety of dynamic settings. The findings of this research hold the potential to impact numerous domains, advancing the field of human presence detection and facilitating more intelligent and responsive environments.

The remainder of this thesis is structured as follows: Section 2 provides a theoretical framework for the study, including the fundamentals of radar and IR-UWB technology, and fundamental ML concepts. Section 3 presents the methodology employed for the study, including the experimental setup, data collection, ML architectures deployed, and performance evaluation. Section 4 describes the preprocessing steps and the final dataset used for training and evaluation, including the data collection process and the labeling procedure. Section 5 presents the findings of the experiments, including discussions about the performance of the ML models and the impact of different preprocessing steps. Section 6 concludes the thesis and discusses potential future work.

1.2 Related work

Little literature was found on the use of a dual-antenna IR-UWB radar sensor for PC aside from the work of Nguyen et al. for localization and detection [8]. But this work did not include the use of ML models for PC. Thus, we will be reviewing the literature on the use of a single-antenna IR-UWB radar sensor for PC. In almost all the papers reviewed, NOVELDA's X4M03 low-power UWB radar [11] has been used.

Radar-based PC using ML is a relatively new field of research, but it has already shown promising results compared to early radar-based PC approaches [7]–[10]. Choi et al. proposed an algorithm based on a strategy of understanding the pattern of the received signal according to the number of people [7]. They generated a probability-density function (PDF) based on the amplitudes of the main pulses from the major clusters found and derived a maximum likelihood equation resulting in a mean absolute error (MAE) of 0.68 for PC up to nine people. Nguyen et al. proposed a filtering method using a Kalman filter followed by a modified CLEAN algorithm and a target localization and tracking step [8]. For a two-person localization and tracking scenario, they achieved detection rates of 73% and 87% for each person respectively.

Recent studies have attempted to apply ML-based approaches to radar-based PC [12]–[18]. Such efforts differed from the traditional use of scattering center extractions (SCE) algorithms, which involves identifying and extracting information about the scattering centers present in the radar return signal, and rather consisted of a set of handcrafted features which a data-driven classifier uses for predicting the correct class, for example the number of people present.

Bao et al. utilized handcrafted features from multi-scale range-time maps and used these to classify the number of people using a CNN model [18]. Their proposed features and prediction on a number of people from zero to 10 people resulted in an average of 61.5%, where the lowest accuracy of 7.6% was for the class of seven people in the detection zone, and their highest of 100% was for zero people present in the detection zone. This study was conducted using a single-antenna radar sensor. Whereas in our research, we will be using a dual-antenna radar sensor.

Choi et al. have done significant research in robust detection of presence of individuals in indoor environments, and radar-based PC both with and without using deep learning (DL) [7], [13]–[17]. In a more recent paper [17], they proposed a preprocessing pipeline to transform the raw return radar signals into a better matched form for a DNN. Furthermore a novel ML architecture that reflected the spatiotemporal characteristics of the signals was designed. The MAE for both indoor and outdoor environments were 0.011 and 0.088 respectively for a multi-class classification problem from zero to 10 people. The study was conducted using the same single-antenna radar sensor as Bao et al. [18]. The results from the paper indicates that their proposed preprocessing pipeline and ML architecture is robust to both indoor and outdoor environments.

Stephan et al. [19] applies ML models for PC, but uses the frequency-modulated continuous wave (FMCW) radar consisting of one transmitter antenna and three receiver antennas. The proposed method involves training the model on supervised RD maps and knowledge distillation for data from a different radar sensor with different position and orientation. But during evaluation, only

the RD maps were used. The results from the paper, where the classification problem was from zero to six people, scored an average accuracy of 0.7143. However, for the predictions from two to six people, their model scored an average accuracy 0.6476.

The reviewed literature shows that the use of ML for PC on IR-UWB radar sensors is a promising approach, and that it has the potential for use in real-world applications. However, the power consumption provided by the single-antenna radar sensors used in the reviewed offered typically 20 mW. With the new Ultra-low Power Presence Sensor by NOVELDA, the power consumption is reduced to below 100 μW, which is a significant improvement. The sensor also offers a dual-antenna radar system, which allows for two-dimensional spatial information: range and direction-of-arrival in azimuth angle. This opens up for new possibilities in the field of radar-based PC and tracking.

2 Theoretical Framework

The theoretical framework in this section will explain the relevant theory behind different concepts and methods used in this research project. More specifically, this section will be divided into two sections: the first section provides the framework for radar systems, more specifically IR-UWB radars, while the last section covers relevant topics within deep learning.

2.1 Radar systems

The radio detection and ranging, also known as radar, is a radiolocation system that involves emitting electromagnetic (EM) waves and measuring the time it takes for the waves to reflect back from any surrounding objects. By analyzing the time delay and characteristics of the reflected waves, radar systems can determine the range, angle, and velocity of the objects.

2.1.1 Radar range equation

The radar range equation is a fundamental equation in radar systems that describes the relationship between the transmitted power, the received power, and the range of the target. The radar range equation is given by Equation 1 and the derivation can be found in Chapter 2 from *Principles of Modern Radar: Basic Principles* [20].

$$P_{\mathrm{r}} = \frac{P_{\mathrm{t}} G_{\mathrm{t}} G_{\mathrm{r}} \lambda^2 \sigma}{(4\pi)^3 R^4} \tag{1}$$

From Equation 1, we can see that the received power P_{r} is inversely proportional to the fourth power of the range R. This means that the received power P_{r} decreases rapidly with increasing range R. This is known as the radar range loss, and is the main reason why radar systems are limited in range.

2.1.2 Pulse radar principles

The radar system used in this research project is an IR-UWB radar system, which is a type of radar system that utilizes ultra-short and low energy pulses to accurately determine the range and velocity of objects. Combined with the high range resolution, low power consumption, and low cost, IR-UWB technology provides a cost-effective and reliable solution for a wide range of indoor applications, such as room occupation detection, object tracking and security systems.

The basic principle of IR-UWB is based on the measurement of time of flight τ between the transmission of the impulse signal and the reception of the reflected signal from the target. If d is the distance to the target and c is the speed of light, τ is given by:

$$\tau = \frac{2d}{c} \tag{2}$$

Thus, the distance d between the radar and the target can easily be found by solving Equation 2 for d.

The high range resolution was previously mentioned as one of the advantages of IR-UWB radars. The range resolution ΔR of a radar is the minimum separation between two targets that can be distinguished and is given by the speed of light c multiplied by the pulse width τ_{p} of the signal divided by two as shown in Equation 3:

$$\Delta R = \frac{c\tau_{\mathrm{p}}}{2} \tag{3}$$

where τ_p is the pulse width of the signal. From Equation 3, we can see that the range resolution ΔR is inversely proportional to the pulse width τ_p, meaning that the smaller the pulse width, the higher the range resolution. Typical range resolutions for IR-UWB radars are in the order of centimeters.

2.1.3 Pulse-Doppler signal processing

In Pulse-Doppler signal processing, measurements and processing of Doppler data can be divided into two domains: "fast-time" and "slow-time", which can be considered to be the range dimension and the pulse number dimension. The fast-time/slow-time matrix of coherent, complex baseband data is shown in Figure 1.

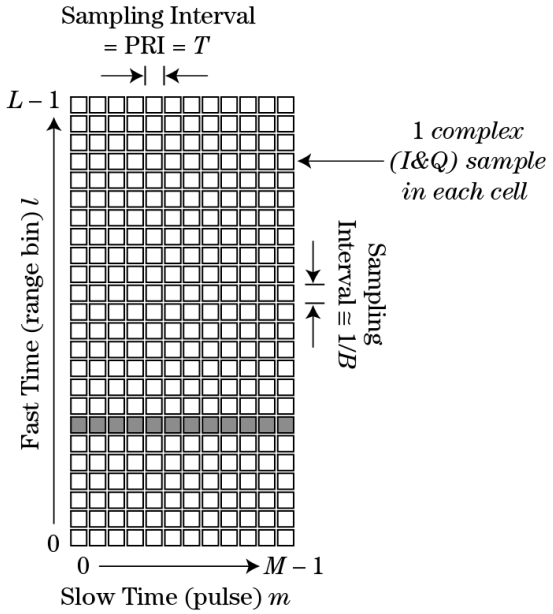


Figure 1: Illustration of the two-dimensional pulse-Doppler data. Reprinted from [20].

Fast-time is associated with the time scale over which a single radar pulse and its corresponding echo are transmitted and received, and is used for capturing range measurements. It is important that the sampling rate in fast-time is high enough to capture the full bandwidth of the signal, which is at least equal to the fast-time signal bandwidth B. B is given by the reciprocal of τ_p as shown in Equation 4. Each sample corresponds to a range bin, and the index of the sample can be directly related to the range of the target from the radar.

$$B = \frac{1}{\tau_p} \tag{4}$$

Slow-time is associated with the time scale over which a series of pulses are transmitted and received and is used for capturing Doppler measurements. The slow-time is sampled at the pulse repetition interval (PRI) T, which is the time between the start of one pulse and the start of the next pulse. Thus, the sampling rate in slow-time is given by $\frac{1}{T}$, also known as the pulse repetition frequency (PRF). Each row in the matrix of Figure 1 corresponds to a series of measurements from the same range bin over M successive pulses.

In the context of slow-time, the Doppler shift f_d is the change in frequency of the reflected signal from the target due to the relative motion between the radar and the target. f_d is given by Equation 5:

$$f_d = \frac{2v}{c} f = \frac{2v}{\lambda} \tag{5}$$

where f and λ are the transmitted frequency and wavelength, v is the radial velocity of the target, and c is the speed of light. From Equation 5, we can see that f_d is directly proportional to v. This means that a positive value of v indicates a positive Doppler shift.

Pulse-Doppler processing analyzises the spectrum of the slow-time data for each range bin, resulting in a two-dimensional matrix of range-frequency-power referred to as range-Doppler matrices. These are generated by applying a discrete fourier transform (DFT) to each range bin. Figure 2 shows an example of a fast-time/slow-time matrix converted to a range-Doppler matrix. In practice, the range-Doppler matrix is generated by applying a fast-fourier transform (FFT) as it is more computationally efficient.

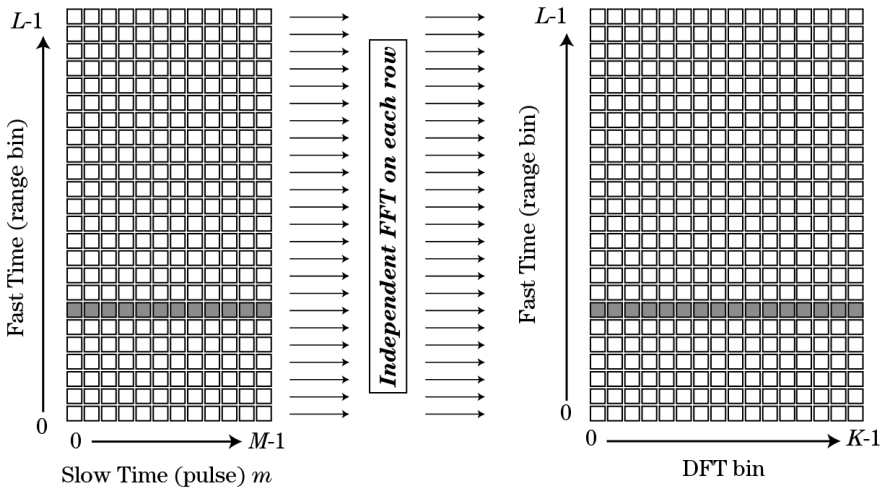


Figure 2: Illustration of the range-Doppler matrix. Reprinted from [20].

In the new two-dimensional data matrix, the range bins are represented by the rows and the Doppler bins are represented by the columns. The value at a specific cell in the matrix represents the power of the signal at a specific range and Doppler frequency, making it possible to distinguish between different targets at different ranges and velocities. One of the advantages of using range-Doppler matrices is that it is possible to distinguish between stationary and moving targets given that stationary targets result in a Doppler frequency of zero.

With an IR-UWB radar, the resolution in range and Doppler are fine enough to capture the movements of the human body both when walking or standing still. This is due to the fact that it is almost impossible for the human body to stand still [21]. In the context of people counting, this means that it is possible to distinguish between people and stationary objects such as furniture and walls. However, a limitation arises from the fact that a single antenna typically has a broad beamwidth, and therefore, multiple targets at different azimuth and elevation but same range might be represented as a single combined target.

A method of overcoming this problem is to employ multiple receiver antennas in an array, which can be used to form a narrow beam in a desired direction. This is known as beamforming. Digital beamforming is a method of combining the signals from multiple receivers digitally to form a narrow beam that is pointed in a desired direction. This is done by adjusting the phase shift $\Delta\phi$ and summing the signals from each receiver channel such that the signals are in phase in the desired angle and out of phase in others. This effectively sums the signals constructively in the desired angle and destructively at other angles. The narrowness of the beam is determined by the number of receiver channels and the distance between the antennas, indicating that a better angular resolution comes at the cost of a larger antenna array.

2.1.4 Limitations of radar systems

In Section 2.1.1 it was mentioned that the received power P_r decreases rapidly with increasing range R. This is only one of the limitations of radar systems. Other limitations include clutter and noise from unwanted echoes due to i.e. stationary objects and walls. In environments with many reflective surfaces, such as indoor environments, the radar signals can reflect multiple times before reaching the receiver, resulting in a phenomenon known as multipath effects. This can cause ghost targets to appear at incorrect positions, due to the time of flight delay, which can be difficult to distinguish from real targets. Lastly, if a small moving object is located behind a larger stationary or moving object, where there is non-line-of-sight (NLOS) between the radar and the smaller object, the radar might be unable to detect the smaller object.

The limitations mentioned above are all challenges that must be taken into account when designing a radar system, i.e. in terms of the antenna arrangement, the location of the radar system, and the environment in which the radar system is used.

2.2 Machine Learning

Machine learning (ML) is a subfield of artificial intelligence (AI) that focuses on the development of algorithms that can learn from data such as numbers, images, and text, and make predictions on new data. This gives the computer the ability to learn how to solve a problem without being explicitly programmed to do so. A research brief from the Massachussetts Institute of Technology (MIT) [22] breaks down the functions of a ML system into three categories: descriptive, predictive and prescriptive. Descriptive analytics is the most basic form of ML, where the system is used to describe what has happened in the past. Predictive analytics is a more advanced form of ML, where the system is used to predict what will happen in the future. Prescriptive analytics is the most advanced form of ML, where the system is used to prescribe what action should be taken in the future.

2.2.1 Learning problems

Machine learning algorithms can broadly be categorized into three groups of problems: supervised, unsupervised, and reinforcement learning.

Supervised learning. Supervised learning is a class of machine learning where during training, the algorithm is training on a tuple of (X, y), where X are the input data and y are the corresponding labels. The goal of the algorithm is to learn a mapping function $f : X \rightarrow y$ such that when introduced to only X, the algorithm can correctly map it to the output y. This is done by minimizing a loss function $L(y, f(x))$ that measures the difference between the predicted output $f(x)$ and the true output y.

There are two main types of supervised learning problems: regression and classification. In regression problems, the output y is a continuous value, while in classification problems, the output y is a discrete value representing a certain class. Supervised ML requires labeled input data during the training phase, which means that the data must be annotated with the corresponding labels for the algorithm to train correctly. Such work is often done manually by a human, which can be a time-consuming and expensive process.

Unsupervised learning. Unsupervised learning does not use labeled data during training, but instead, the algorithm is solely dependant on the input data X. The goal of the algorithm is to learn the underlying structure of the data, such as patterns and relationships, without any prior knowledge of the data. Unsupervised learning is typically used for clustering, which involves grouping data points into clusters based on their similarities, dimensionality reduction, which involves reducing the number of features in the data based on their importance, and anomaly

detection, which involves identifying data points that are significantly different from the rest of the data.

Reinforcement learning. Reinforcement learning is a class of machine learning where the algorithm learns by interacting with its environment. The algorithm is rewarded for performing the correct action and penalized for performing the wrong action. The goal of the algorithm is to learn the optimal policy, which is a mapping from the state of the environment to the action that maximizes the reward. Reinforcement learning is typically used for robotics, gaming, and navigation.

To decide for which algorithms to use requires a thorough understanding of the problem at hand. This involves understanding the data that is available, the desired output, and the desired performance. Real life problems are often complex and the data is not annotated or correctly formatted, which means that it is often necessary to apply different preprocessing techniques before the data can be used for its intended purpose.

2.2.2 Perceptron and multi-layer perceptron

Neural networks are a subset of machine learning that are inspired by the structure and function of how the human brain processes information. The basic building block of a neural network is the perceptron, which is a mathematical model of a biological neuron first proposed by Frank Rosenblatt in 1958 [23]. Figure 3 shows the proposed model of a perceptron.

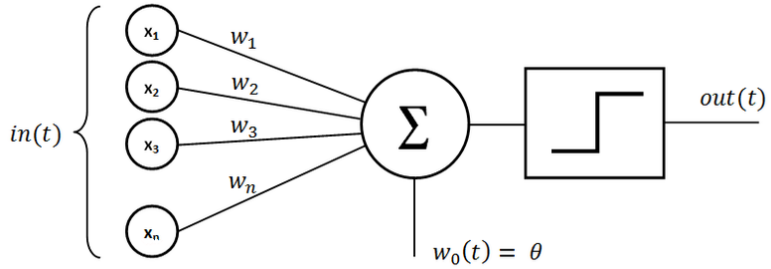


Figure 3: Illustration of an artificial neuron which the perceptron is based on. Reprinted from [24].

Perceptron. The perceptron takes a set of inputs $x_1, x_2, ..., x_n$ and multiplies each input with their respective weights $w_1, w_2, ..., w_n$. The weighted inputs are then summed together with a bias b (denoted as θ in the figure), and passed through a unit step activation function $f(z)$ to produce an output $\hat{y}$ based on the threshold Θ. In mathematical terms, the perceptron can be described by Equation 6:

$$z = \sum_{i=1}^{n} w_i x_i + b \text{ and } \hat{y} = f(z) = \begin{cases} 1 & \text{if } z \geq \Theta \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

A learning algorithm for the perceptron was proposed by Rosenblatt [23] for a binary classification problem. This algorithm is a supervised learning algorithm that adjusts the weights of the perceptron based on the error of the output. The weights are adjusted according to Equation 7:

$$w_i \leftarrow w_i + \Delta w_i \tag{7}$$

where Δw_i is given by Equation 8:

$$\Delta w_i = \eta(y - \hat{y})x_i \tag{8}$$

where η is the learning rate, y is the true label, and $\hat{y}$ is the predicted label. The learning rate η is a *hyperparameter* that controls how much the weights are adjusted during each iteration. The learning rate is typically set to a small value, such as 0.01, to ensure that the weights are not adjusted too much during each iteration.

In short, we can summarize the simplest form of the perceptron as follows: The weights w_i are initialized to small random numbers, for given inputs of an example from the training data, the output $\hat{y}$ is computed, the weights are updated according to Equation 7, and the process of computing $\hat{y}$ and updating the weights are repeated until the algorithm converges and makes no mistakes. The perceptron algorithm is a linear classifier, which means that it can only classify linearly separable data.

The perceptron algorithm was a breakthrough in the field of machine learning, but it was limited to linearly separable data. This limitation has later been overcome by the introduction of the multi-layer perceptron (MLP). The MLP is a feedforward neural network (FFNN) that consists of an input layer, one or more hidden layers in the middle, and an output layer. Figure 4 depicts an example of an MLP with two hidden layers. The outputs of each neuron in a layer $i-1$ is fed as input to all the neurons in layer i.

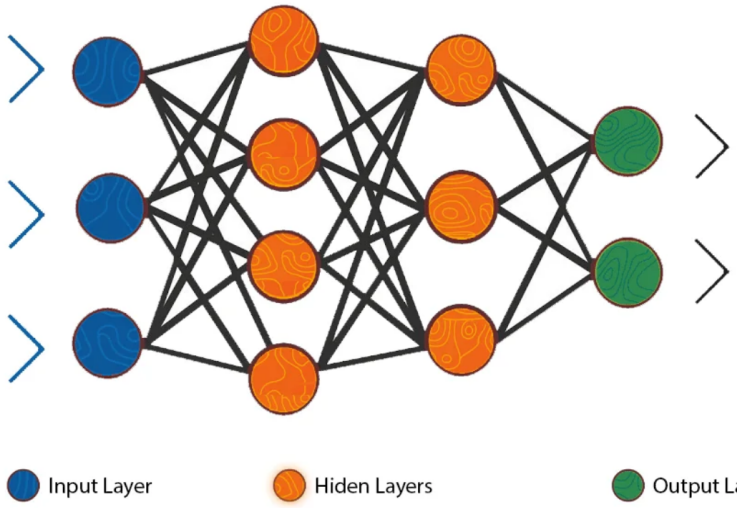


Figure 4: Illustration of a multi-layer perceptron with one hidden layer. Reprinted from [25]

MLP. The MLP is a universal function approximator, which means that it can approximate any function given a sufficiently large number of hidden layers and neurons. The MLP is trained using the backpropagation algorithm, which is an algorithm for supervised learning of MLPs. The backpropagation algorithm is a generalization of the delta rule for the perceptron, and is used to calculate the gradient of the loss function with respect to the weights. The weights are then updated using gradient descent. With the backpropagation algorithm, the requirements for the activation function is that it must be differentiable, which the unit step function is not. Hence, the introduction to many more popular activation functions which are commonly used in modern neural networks.

2.2.3 Activation functions

Activation functions are used to introduce non-linearity into the neural network. Without activation functions, the neural network would be a linear regression model, which is a linear classifier. This means that the neural network would not be able to learn complex patterns in the data. As we say for the perceptron, Rosenblatt proposed the unit step activation function, but this activation function was limited to binary classification problems and was not differentiable. Since then, many different activation functions have been proposed. The most popular ones are sigmoid, tanh, and

rectified linear unit (ReLU)

Sigmoid. The sigmoid squashes the input into the range [0, 1] and is given by Equation 9:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{9}$$

where z is the input to the sigmoid function. Although the sigmoid is differentiable, the activation function is prone to the vanishing gradient problem, which is a problem that occurs when the gradient of the loss function approaches zero. This causes the weights to be updated very slowly, which can lead to the neural network not learning anything at all.

Tanh. The tanh squashes the input into the range [-1, 1], making it zero-centered. The tanh function is given by Equation 10:

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{10}$$

where z is the input to the tanh function. The tanh is often prefered over the sigmoid function in practice. However, the tanh function is also prone to the vanishing gradient problem.

ReLU. The ReLU is the most commonly used activation function and is defined as the positive part of its argument:

$$\text{ReLU}(z) = \max(0, z) \tag{11}$$

From Equation 11, z is the input to the ReLU function. The ReLU function is not prone to the vanishing gradient problem, but it is prone to the dying neuron problem, which is a problem that occurs when the gradient of the loss function is zero. This causes the neuron to be stuck in a state where it does not activate and does not contribute to the learning process. Although the ReLU function is not differentiable at $z = 0$, in practice, it is differentiable at all other points, which is sufficient for the backpropagation algorithm to work. In addition, the activation function is computationally efficient, which makes it a popular choice for many neural networks.

Softmax. The softmax is a generalization of the sigmoid function that is used for multi-class classification problems. It calculates the probabilities of each class over all possible classes and is used as the final layer in a neural network to output the probabilities of each class.

2.2.4 Optimizers

Optimization algorithms are used to minimize a loss function $L(\theta)$ by iteratively moving in the direction of the negative gradient. There are many different optimization algorithms, and the choice of algorithm depends on the problem at hand. The most popular ones are gradient descent, stochastic gradient descent (SGD), momentum, Adaptive gradient algorithm (AdaGrad), Root mean square propagation (RMSProp), and adaptive moment estimator (Adam).

Gradient descent. The gradient descent is the first and simplest optimization algorithm for ML and neural networks training. The gradient is the vector of partial derivatives of the loss function with respect to the parameters θ. The parameters θ are updated according to Equation 12:

$$\theta \leftarrow \theta - \eta \nabla_{\theta} L(\theta) \tag{12}$$

where η is the learning rate, which is a hyperparameter that controls how much the parameters are adjusted during each iteration. The learning rate is typically set to a small value, such as 0.01, to ensure that the parameters are not adjusted too much during each iteration.

SGD. The SGD is a variant of the gradient descent algorithm. The difference between gradient descent and SGD is that in SGD, the gradient is computed for a single example at a time, while in gradient descent, the gradient is computed for the entire training set. This makes SGD much faster than gradient descent, but also prone to noise and diverging from the optimal path to the global minimum. The parameters θ are updated according to Equation 13:

$$\theta \leftarrow \theta - \eta \nabla_\theta L(\theta; x^{(i)}; y^{(i)}) \tag{13}$$

where $x^{(i)}$ and $y^{(i)}$ are the input and output of the ith example in the training set.

Momentum. The Momentum accelerates the convergence of the algorithm by adding a fraction γ of the update vector of the past time step to the current update vector. The parameters θ are updated according to Equation 14:

$$\begin{aligned} v_t &\leftarrow \gamma v_{t-1} + \eta \nabla_\theta L(\theta) \\ \theta &\leftarrow \theta - v_t \end{aligned} \tag{14}$$

where v_t is the velocity vector at time step t, γ is the momentum term, and η is the learning rate.

AdaGrad. The AdaGrad optimizer adapts the learning rate η for each parameter θ_i at time step t based on the past gradients that have been computed for θ_i. The parameters θ are updated according to Equation 15:

$$\theta \leftarrow \theta - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t \tag{15}$$

where G_t is the sum of the squares of the gradients up to time step t, and ϵ is a small constant to prevent division by zero.

RMSProp. The RMSProp optimizer adapts the learning rate η for each parameter θ_i at time step t based on the past gradients that have been computed for θ_i. The parameters θ are updated according to Equation 16:

$$\theta \leftarrow \theta - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \odot g_t \tag{16}$$

where $E[g^2]_t$ is the exponentially decaying average of past squared gradients, and ϵ is a small constant to prevent division by zero.

Adam. The Adam optimizer combines the advantages of two other extensions of stochastic gradient descent: AdaGrad and RMSProp. The parameters θ are updated according to Equation 17:

$$\theta \leftarrow \theta - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{17}$$

where $\hat{m}_t$ and $\hat{v}_t$ are the first and second moment estimates of the gradients, and ϵ is a small constant to prevent division by zero.

2.2.5 Performance metrics

Performance metrics are used to evaluate the performance of a ML model. The choice of performance metric depends on the specific task one is trying to solve. For our case, we have a multi-class classification problem, and for these classification problems, the most common performance metrics are confusion matrix, accuracy, precision, recall, and F1-score. Although confusion matrices are not a metric, it is still a fundamental tool for computing the other metrics and for visualization.

Confusion matrix. The confusion matrix is a table that is used to visualize the performance of a classification model. For a n-class classification problem, the confusion matrix is a $n \times n$ matrix that contains the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Figure 5 shows an example of a confusion matrix for a binary classification problem.

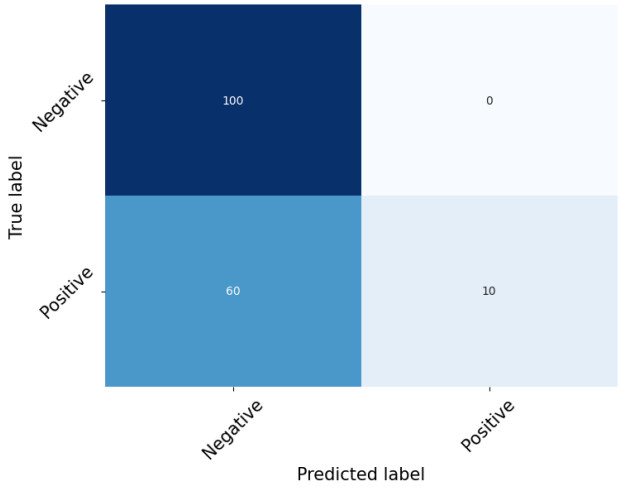


Figure 5: Illustration of a confusion matrix for a binary classification problem.

Accuracy. The accuracy is the most common performance metric for classification problems. It is defined as the ratio of the number of correct predictions to the total number of predictions. Although widely used, accuracy is not always the best metric, especially when the data is imbalanced. For example, if the data contains 90% of class A and 10% of class B, a model that always predicts class A will have an accuracy of 90%, even though the model is not able to predict class B at all. In such cases, accuracy is not a reliable indicator of the model's performance. This is where precision and recall offer more insight into the model's performance.

Precision. The precision is used as a metric when the goal is to minimize the number of FP. It is defined as the ratio of the number of TP to the total number of positive predictions:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{18}$$

Precision is useful in scenarios where the cost of FP is high and the cost of a FN is low. From Figure 5 the precision can be seen as the ratio of the number of TP to the sum of TP and FP. The weakness of the precision metric is that it does not take into account the number of FN. For example, given the confusion matrix in Figure 5, the precision will give a value of 1.0. This means that 100% of the predicted positives were correct. However, the metric gives no insight into how many of the TP that were correctly classified.

Recall. The recall is used when the goal is to minimize the number of FN. It is defined as the ratio of the number of TP to the total number of actual positives:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{19}$$

From the same confusion matrix, the recall would result in a value of 0.14. This means that only 14% of the actual positives were predicted correctly. It is possible to train a model to focus entirely on precision or recall, but in cases where you want the model to both have a high recall and a high precision, the F1-score is a good metric to use.

F1-score. The F1-score is the harmonic mean of precision and recall, and optimizing for the F1-score is equivalent to optimizing for both precision and recall. The F1-score is defined as the ratio of the product of precision and recall to the sum of precision and recall:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{20}$$

2.2.6 Convolutional neural networks

Convolutional neural networks (CNN) are a type of artificial neural networks (ANN) that are commonly used for image classification and object detection. The architecture make use of convolutional layers, which is a layer that applies a convolution to the input. The convolution is a mathematical operation that is specifically designed for processing pixel data. Convolutions involve using a small matrix with learnable weights, known as kernels or filters, these filters are usually a square matrix with odd dimensions. The filter is convolved with the input by sliding the filter over the input and computing the dot product between the filter and the input at each position. The filter is then shifted by a certain number of pixels, known as the stride, and the process is repeated until the entire input has been covered. The stride is a hyperparameter that must be specified before training the model. The stride is typically set to 1, which means that the filter is shifted by one pixel at a time. Figure 6 shows an example of a convolutional operation with a 3×3 filter and a stride of 1 over an input of size 5×5 for the first row, effectively shrinking the input to a feature map of size 3×3.

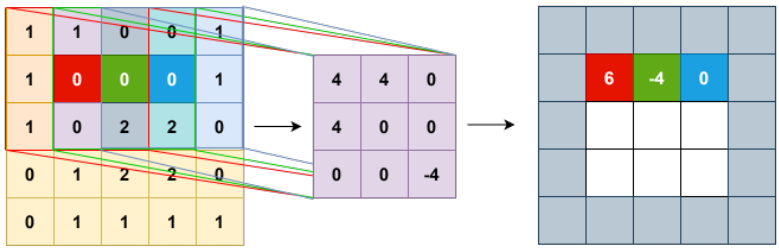


Figure 6: Illustration of a convolutional operation with a 3×3 filter and a stride of 1 over an image of size 5×5.

The feature map is a matrix of values that represents the presence of certain features in the input. The size of the feature map is determined by the size of the input, the size of the filter, and the stride, where in this case we get a 3×3 feature map as an output after the convolution. After a convolution, an activation function, often a ReLU, is applied to the feature map to introduce non-linearity. The output of the activation function is then passed on to the next layer.

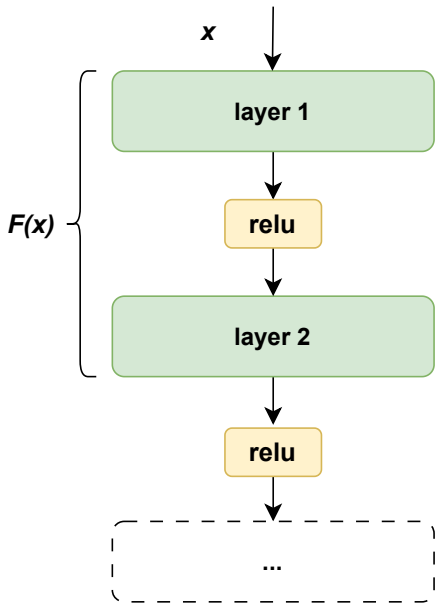
Usually for a convolutional layer, multiple filters are used to produce multiple feature maps. The number of filters is a hyperparameter that must be specified before training the mode and determines the depth of the layer. Each of these filters have learnable weights, which means that the

filters are learned during training to locally look for different features such as edges, textures or colors. By stacking convolutional layers, the network can learn to detect more complex patterns and features.

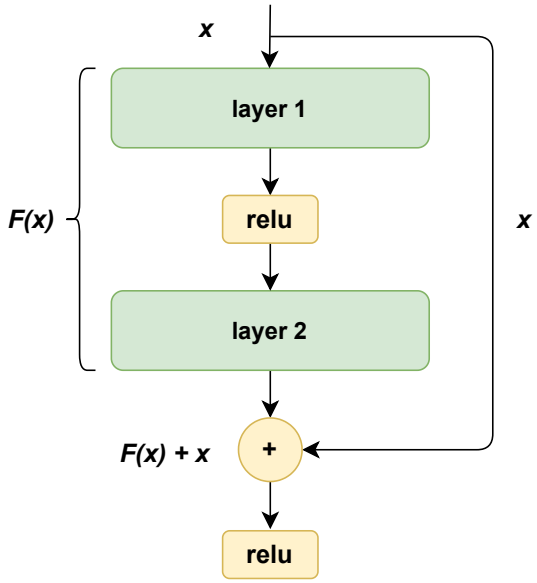
2.2.7 Residual neural networks (ResNet)

ResNets are a type of CNN that are designed to be deeper than previous CNNs. The architecture was introduced to address the problem of vanishing gradients, which usually hinders the training of very deep networks.

One of the key properties of ResNets is that they use residual connections, also known as skip connections, that allow the network to directly propagate information from earlier layers to later layers. These skip connections allows the ResNet to learn residual mappings, which capture the difference between the desired mapping and the current representation. The skip connections act as a bypass mechanism, allowing the network to preserve and reuse information from earlier layers. Figure 7a illustrates how the output of layer 1 propagates to layer 2, and Figure 7b illustrates how the skip connection works by adding the input of layer 1 to the output of layer 2.



(a) Weight propagation in a CNN model.



(b) Weight propagation with residual blocks in a ResNet model.

2.2.8 Recurrent neural networks (RNN)

RNNs are a type of ANN that are commonly used for sequential data, such as time series data, videos and audio. In contrast to traditional feedforward neural networks, RNNs have feedback loops that allows information to persist within the network, also known as hidden states or memory cells. This makes it possible for the network to maintain a memory of previous inputs when processing the current input. The simplest RNN can be seen as a repetition of a single memory cell where a cell combines the current input x_t and the previous input state h_{t-1}, applies an activation function (usually tanh), and produces the output h_t. This output is then fed back into the cell as the input state for the next iteration, but also as the output of the cell. Figure 8 shows an illustration of a RNN with a single memory cell while Equation 21 shows the mathematical formulation of the RNN for the current timestep.

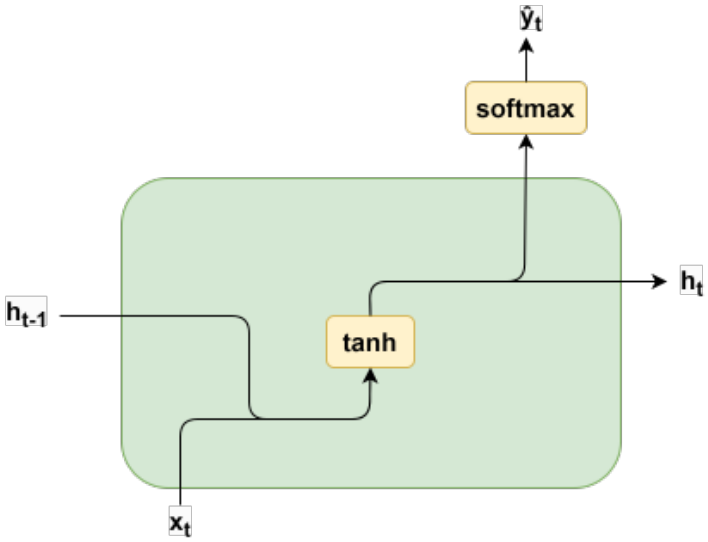


Figure 8: Illustration of a RNN with a single memory cell.

$$h_t = \tanh(W_h h_{t-1} + W_x x_t + b) \tag{21}$$

When training a RNN, a variant of backpropagation, known as backpropagation through time (BPTT), is used. BPTT calculates the loss at every timestep and accumulates the gradients over all timesteps. The gradients are then used to update the weights of the network at every timestep using an optimizer. Due to the activation function used, RNNs are also prone to the vanishing gradient problem. This is often a problem when training RNNs on long sequences, where the gradients either become too big (exploding gradient), or too small (vanishing vanish). This makes it difficult for the network to learn long-term dependencies. New architectures have been proposed to improve the performance and counter these problems. The most popular variants is the long short-term memory (LSTM) cell and the gated recurrent unit (GRU) cell.

LSTM is a variant of the RNN where the LSTM cell has a cell state and three types of gates: input gate, forget gate, and output gate. The cell state is passed from the input to the output of the cell which allows the LSTM to learn long-term dependencies. The input gate controls which information should be added to the cell state, the forget gate controls how much of the long-term memory should be forgotten or discarded from the cell, and the output gate determines which part of the cell state builds the output. All three gates are controlled by their respective sigmoid activation functions along with their respective weights and biases.

These gating units are Figure 9 shows an illustration of a LSTM cell.

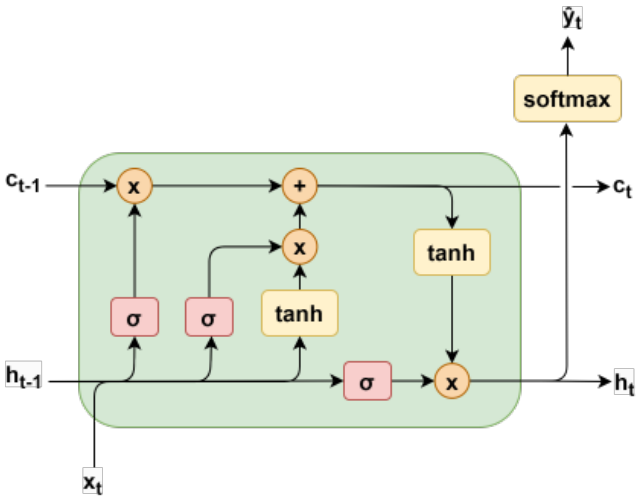


Figure 9: Illustration of a LSTM cell.

GRU is a variant of the RNN where the GRU cell consists of only two gates: reset gate and update gate. The reset gate controls how much of the past information that should be forgotten, and the update gate, similar to the input and forget gate of the LSTM cell, controls what information to forget, and what new information to add. Figure 10 shows an illustration of a GRU cell.

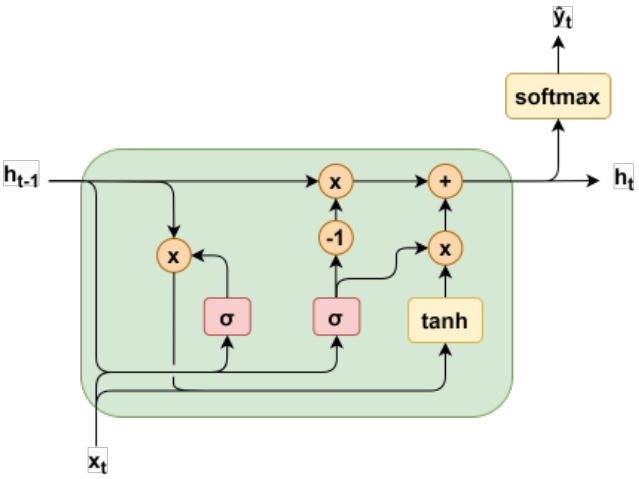


Figure 10: Illustration of a GRU cell.

Both LSTM and GRU are both popular types of RNNs used for processing sequential data. They are both designed to tackle the vanishing gradient problem that the basic RNN was suffering from. However, there is no clear winner between the two, as they both have been reported in the literature to perform equally well in most cases, or one outperforming the other in other cases. As GRU has a less complex structure than LSTM, it is often faster to train and performs better on smaller datasets [26]. However, LSTM have the advantage of having a cell state that allows the network to learn long-term dependencies, making it more suitable for long sequences. It is often recommended to try both LSTM and GRU when training a RNN, and choose the one that performs best.

3 Methodology

3.1 Experimental setup

In this section we describe the experimental environment and setup used for recording the dataset and labeling for our People Estimation (PE) efforts. We will also explore the dataset and describe the data processing pipeline used to extract the features from the dataset.

The experiments are carried out in a controlled indoors environment at NOVELDA's main office in Oslo. The area of interest is a 4×4 m square, which is further divided into a grid of 16 squares of 1 m^2. The experimental setup consists of NOVELDA's novel Ultra-low Power Presence Sensor, which is a dual-antenna IR-UWB radar module, mounted on an 'Adam' ArUco box 1.5 m above the ground, two Kinect RGB cameras mounted to the ceiling for validation and labeling, a laptop for recording and storage, and duct tape for marking the areas of interest. The experimental setup is illustrated in Figure 11.

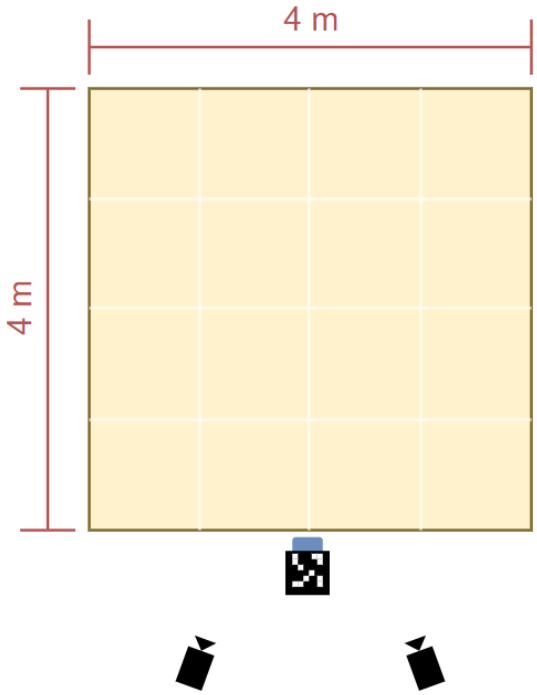


Figure 11: Illustration of the experimental setup.

3.1.1 NOVELDA Ultra-low Power Presence Sensor

The radar used for this study is the newly showcased NOVELDA Ultra-low Power Presence Sensor [27]. This radar sensor is a high-performance, low-power, dual-antenna IR-UWB radar with the world's lowest power consumption of below 100 μW. The radar sensor features a complete sensor module with antennas and on-chip processing within 5×30 mm, and is capable of reliably detecting breathing motion at 10 m from the sensor.

Figure 12: NOVELDA Ultra-low Power Presence Sensor. Reprinted from [28].

3.1.2 Data collection

The participants in every recording are following a manuscript, describing the activity performed. These manuscripts are provided such that participants in every recording follows the same procedures, making the recordings reproducible, identifiable and comparable. Additionally, the manuscripts aim to address worst-case scenarios where radars are known to have difficulties in correctly detecting the targets as mentioned in Section 2.1.4. Such cases happen for instance when two or more subjects are walking at the same speed and range from the radar, which makes the radar unable to distinguish between the subjects in the RD domain. Every recording is provided with a unique ID. The manuscripts are provided in Appendix A.

After a completed recording, the Kinect RGB camera recordings and raw radar sensor data are stored in a cloud database and are available for download and further preprocessing. The files are stored in a folder with the ID of the recording along with metadata such as the date and time of the recording, the duration of the recording, the settings of the sensor, the number of subjects in the recording, and the manuscript of the recording.

3.2 Machine learning

For our model architecture, it will be optimal to take advantage of the spatio-temporal information from the recordings, given our sequential RD maps. A common approach to take advantage of the spatio-temporal information is to use CNN for feature extraction, followed by a RNN layer, typically GRU or LSTM, for classification on the sequences. Further, we explore the ResNet and the ResNet+RNN architecture.

3.2.1 Baseline CNN

We will employ a CNN model for the task of multi-class classification in our people counting problem using RD maps. The CNN architecture is well-suited for extracting hierarchical representations and capturing spatial dependencies within image-like data, making it an ideal choice for analyzing RD maps. Additionally, due to our three-channel RD maps, we can use the basic 2D CNN architecture for feature extraction, as opposed to the more complex 3D CNN architecture.

The CNN model consists of several key components, including convolutional layers, activation functions, pooling layers, and fully connected layers. These components collectively enable the model to learn discriminative features from the RD maps and make accurate predictions.

The convolutional layers play a crucial role in capturing local patterns and features from the input RD maps. By applying a set of learnable filters to small local regions of the maps, the convolutional layers extract relevant features that are important for classification. The depth and width of the convolutional layers can be adjusted based on the complexity and variability of the RD maps.

Following the convolutional layers, activation functions such as ReLU are applied element-wise to introduce non-linearity into the model. This non-linearity allows the model to capture complex relationships between features and enhance its capacity to learn intricate patterns from the RD maps.

Pooling layers, typically in the form of max pooling or average pooling, are used to downsample the feature maps and reduce the spatial dimensions. These layers help to extract the most salient features while reducing computational complexity and preventing overfitting.

3.2.2 ResNet

ResNet is a popular architecture that has shown great performance in various computer vision tasks, including image classification. By employing ResNet, we can leverage its inherent ability to effectively learn deep representations, enabling the extraction of meaningful features from the RD

maps. The depth of the ResNet architecture allows it to capture intricate patterns and nuances in the data, which is essential for accurately classifying the RD maps. Moreover, ResNet addresses the challenge of vanishing gradients through the use of residual connections, enabling the smooth propagation of gradients during training and enabling the successful training of deep networks. By using ResNet, we aim to exploit its advantages in terms of feature learning, model depth, and training stability, with the expectation that it will enhance the accuracy and robustness of our multiclass classification task. Although, it is worth noting that the increased depth of the ResNet architecture also increases the computational complexity and training time, which could be a potential drawback. Furthermore, the increased depth of the model could also increase the risk of overfitting, which would require us to utilize regularization techniques and ensure there is enough training data.

3.2.3 CNN+RNN

The combination of a CNN and a RNN offers unique advantages for capturing the spatiotemporal information in the radar recordings. Each RD frame in the recording could be seen as a time step in a video where each frame is an RGB image with three channels, representing the different beams of the radar. By using a CNN for feature extraction, we can extract relevant features from each RD frame and capture the spatial dependencies between the different beams. The extracted features can then be fed into a RNN layer, which can recognize the sequential patterns between the RD frames. This combination of CNN and RNN enables the model to learn complex spatiotemporal representations from the RD maps. Whether the RNN layer should be a GRU or LSTM layer is something we will explore in our experiments. The CNN+RNN model has demonstrated success in various sequential data tasks, making it a promising choice for analyzing time-series radar data.

3.2.4 Model training and evaluation

For training the model, the data is split into training, validation, and test set. A data generator is used to load the data in batches, which is then fed to the model. The training is performed using the Adam optimizer with a learning rate of 0.001, and a suitable batch size. The models are initially trained for 100 epochs but may train with less epochs as we will add an early stopping with a patience of 15 epochs. The early stopping ensures that the model will stop training if the performance stagnates. The model weights from the best performance with regards to the lowest validation loss and accuracy is saved. To prevent overfitting of the models, dropout layers are used during training. The models are trained on an NVIDIA A100 Tensor Core GPU provided by Google Colab.

For evaluating the performance of the models, the test set is used. At this stage, we evaluate the performance of the models on the test set using the classification report from sklearn. The most representative metric will be the F1-score as it is a weighted average of the precision and recall, which is suitable for our multiclass classification problem.

4 Dataset and Preprocessing

In this section we detail the preprocessing done to the dataset before training the models. NOVELDA has provided us with 361 raw radar recordings ranging from zero to seven participants following a manuscript in a 4×4m grid. The information about the height, weight and gender of the participants in the recordings are provided in Table 1.

Table 1: Participant information.

Participant	**Height (cm)**	**Weight (kg)**	**Gender**
1	200	105	Male
2	183	86	Male
3	175	74	Male
4	175	70	Female
5	168	59	Female
6	168	65	Female
7	173	55	Female

The number of recordings can further be split into 9 different cases, where each case is a combination of the number of participants and the manuscript. The number of recordings for each case is shown in Table 2.

Table 2: People counting dataset.

Manuscript	**Number of samples**
1	187
2	39
3	27
4	62
5	15
6	7
7	8
8	4
9	12
Total samples	**361**

4.1 Preprocessing

Using NOVELDA's in-house signal processing pipelines, digital beamforming creates three beams from the raw radar sensor data, where one beam is directed in $0°$, and two beams in $\pm 20°$ azimuth respectively.

Due to the significantly high frame rate of the radar sensor, the frame rate was reduced from 250 frames per second (FPS) to 4 FPS by decimating the frame rate. The frame rate reduction is done because of the redundant information presented with regards to the application of PC, where a very high frame rate is not necessarily important.

The range resolution will also be reduced under the same principle, where the original range bin size is decimated by a factor of 4, corresponding to $\frac{0.0714\text{m}}{4} = 0.2856\text{m}$. This reduces the computational complexity of the model. Further, as the area of interest is a 4×4m square, the range bins are cropped to the number of bins corresponding to 4 meters, $\frac{4\text{m}}{0.2856\text{m}} = 14$ range bins, and then rounded up to the closest factor of 16.

Pulse-Doppler signal processing with a FFT size of 128 on each beam for each range bin creates three RD maps, where each map is a 2D matrix of size 16×128. The value in each cell represents the reflected absolute power of the target in the corresponding range and Doppler bin.

To address the range loss from the reflected signal from a target, we multiply the received signal power by the inverse of the range loss factor given by Equation 1. We also convert the RD maps from power values to a logarithmic scale as it is beneficial when the dynamic range of the power values are large. Such a conversion compresses the values, making the values easier to interpret. The RD map may contain zero-values, which is why we also add a pseudocount to the signal values as shown in Equation 22. Further we propose to standardize or normalize the RD maps before training.

Standardizing or normalizing the RD maps have been known to reduce the variance in the data and to make the model converge faster. We will be trying both the techniques on a frame-by-frame basis and on a global basis to see which method works better. The standardization is done by subtracting the mean and dividing by the standard deviation of the RD maps, while the normalization is done by subtracting the minimum value of the RD maps, and then divide the difference of the maximum and minimum value.

$$\text{dB} = 10\log_{10}(Power) \tag{22}$$

Frame models

For the baseline CNN and ResNet, which works on batches of frames, the three RD maps for each frame are stacked in the last dimension to form a 3D matrix of shape (16, 128, 3). The frames from all the recordings are then stored as a single tfrecord file, which is a binary file format for storing sequences of records. The tfrecord file is randomly shuffled and split into a train, validation and test set with a ratio of 80%, 10% and 10% respectively with a fixed random seed for training and testing. The total size of the tfrecord file is 6.5 GB and contains 284,043 frames split as shown in Table 3.

Table 3: Initial dataset for frame models.

People count	**Number of frames**			
–	**Train**	**Validation**	**Test**	**Total**
0	99,585	12,449	12,449	124,483
1	94,050	11,756	11,756	117,562
2	22,524	2,816	2,816	28,156
3	11,074	1,384	1,384	13,842

There is a high chance that the frame models would suffer from the class imbalance in the dataset due to the much higher number frames with values of zero and one. To overcome this problem, we propose to balance the training data by undersampling the majority classes such that the number of frames for all classes are equal. The undersampling is done by randomly selecting the same number of frames for each class. The final dataset for training, validation and testing is shown in Table 4.

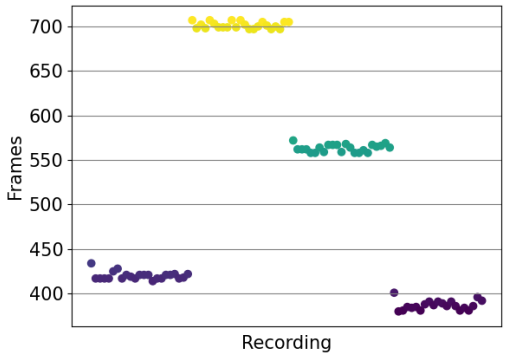
Table 4: Final dataset for frame models.

People count	**Number of frames**			
–	**Train**	**Validation**	**Test**	**Total**
0	11,074	12,449	12,449	35,972
1	11,074	11,756	11,756	34,582
2	11,074	2,816	2,816	16,706
3	11,074	1,384	1,384	13,842

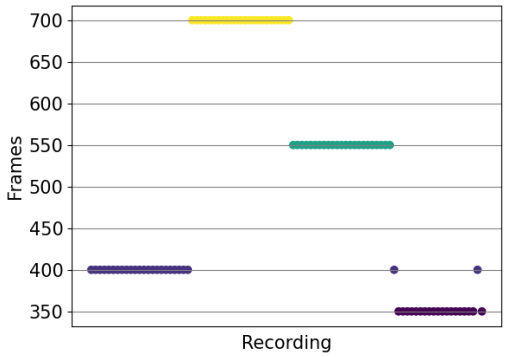
Sequence models

For the sequence model, all three beams for each recording are stored as a separate NumPy array of shape (N, 16, 128), where N is the number of frames in the recording. The recordings with more than three participants are discarded due the lack of data for training the models, giving us 337 recordings for the final dataset.

Further, we extract the number of frames for each recording for trimming or padding due to the variance in recording lengths. Because all recordings start with a 30 second silence period, and end with a 15 second silence period, and knowing that we have approximately four FPS, we have around 180 frames with silence. Thus, trimming and padding is done by removing or adding the first or last N frames from the recordings because valuable information will not be lost. The trimming and padding of the recordings has been done to make the recordings match in length and form batches for training. The fixed lengths have been chosen to be multiples of 50. The decision of whether to trim or pad the recordings is based on the option that results in the least amount of frames being removed or added to the recording. A scatter plot before and after the trimming and padding is shown in Figure 13a and Figure 13b respectively.



(a) Scatter plot of the recording length for each recording.



(b) Scatter plot of the recording length for each recording after trimming or padding.

The processed NumPy arrays are stored in a folder for the processed recordings, where the folder name is the ID of the recording. From Table 2, we can observe that our dataset is suffering from class imbalance even after removing the recordings with more than three participants. Manuscript one and four are drastically more frequent in the dataset than the other manuscripts. To overcome this problem, we decided to balance the dataset by randomly selecting the same number of recordings for each case, which in this case means that the recordings from manuscript one and two are reduced to match the number of recordings from manuscript three. The same is done for manuscripts four, five and six. The final dataset is split in a stratified manner into training, validation and test set with a ratio of 80%, 10% and 10% respectively. The dataset is split such that the recordings from the same manuscript are not split between the sets. The final dataset is summarized in Table 5.

Table 5: Processed dataset.

Manuscript	**Number of recordings**		
–	**Train**	**Validation**	**Test**
1	21	3	3
2	21	3	3
3	21	3	3
4	5	1	1
5	5	1	1
6	5	1	1
Total samples	**78**	**12**	**12**

4.2 Data augmentation

The efforts in creating the dataset is still ongoing and quite recent, thus the size of the dataset is yet not ideal for model training but still holds great value. Given that we are working with RD maps with three channels where each channel represents a "left", "center" and "right" angled beam in azimuth, we propose to swap the "left" and "right" channel. Doing so would give us a new data example with the same label as the original frame, thus augmenting our training data by a factor of two. This technique is illustrated in Figure 14.

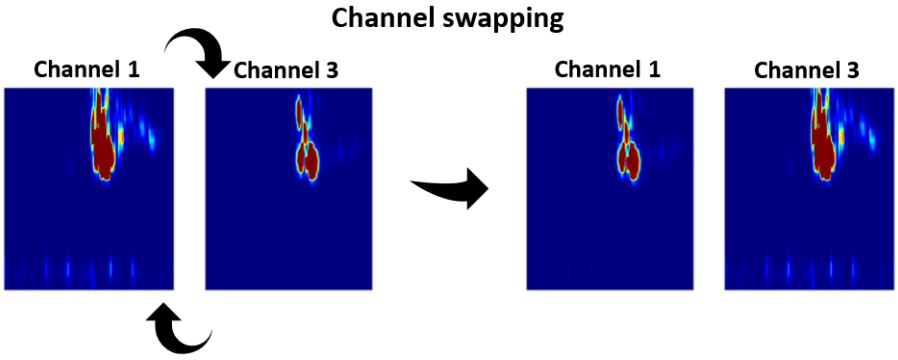


Figure 14: Illustration of swapping channel 1 and channel 3.

4.3 Data labeling

The data labeling was performed by creating a GUI and visualization tool for labeling the radar data. The GUI was created using the PyQt5 framework, which is a Python binding of the cross-platform GUI toolkit Qt. The GUI is used to visualize the radar data as three RD maps, one frame at the time. Using the GUI, the user can load a recording, navigate through the RD frames, and label the radar data by choosing the number of subjects in the frames. The GUI will also show the relative timestamp of each frame in the recording, which is used to synchronize the radar data with the Kinect RGB camera video recordings for validation and ground truth. Figure 15 shows the GUI and visualization tool for labeling the radar data manually.

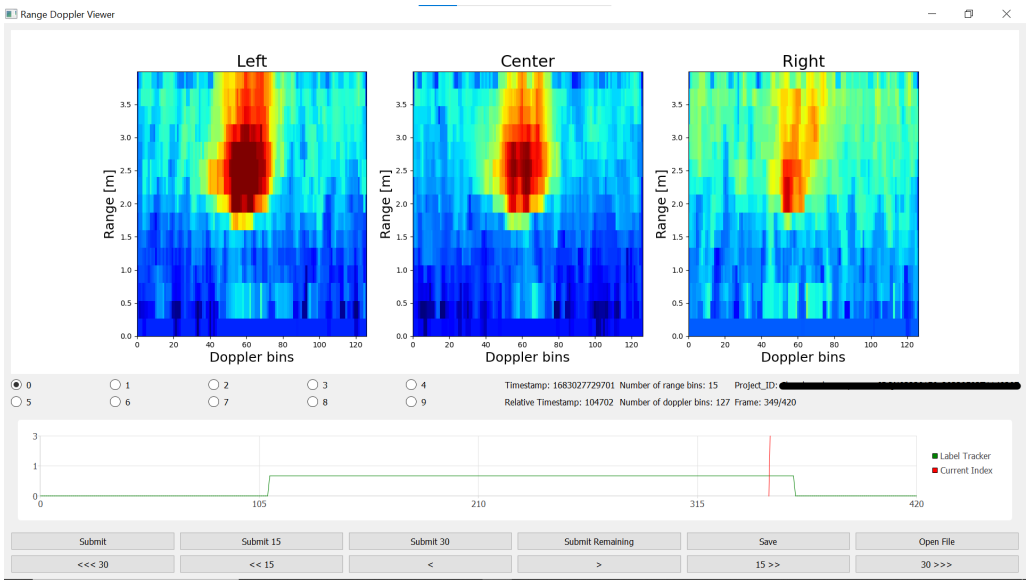


Figure 15: GUI and visualization tool for manually labeling the radar data.

At a later stage, an effort in making an automatic labeling process using the Kinect RGB cameras

for ground truth was completed by NOVELDA's software team. The depth map and position of skeletons from the two cameras were used to extract the position of the subjects in the scene, shown in Figure 16, which was then used to label the radar data. Two cameras were necessary to capture the full scene. Each recording thus had a corresponding label file with the same name as the recording ID. The label file is a JSON file containing a dictionary with metadata, and a list of segments and its value. The value is an integer describing the number of subjects in the scene, while the segments indicates the duration the value is true in milliseconds.

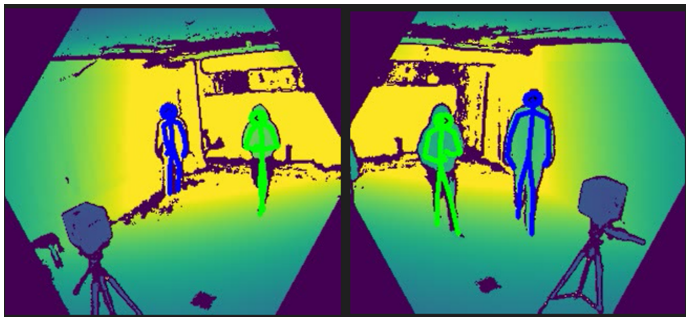


Figure 16: Depth field from the two Kinect RGB cameras showing the full scene being captured by the two cameras.

Furthermore, Figure 17 shows the depth field from the two Kinect RGB cameras showing three participants in the scene. The participants are standing still in the scene, which is the stand still period of manuscript 6. In conventional PC using SCE algorithms, the peaks in the would be overlapping making it difficult to resolve the number of subjects in the scene.

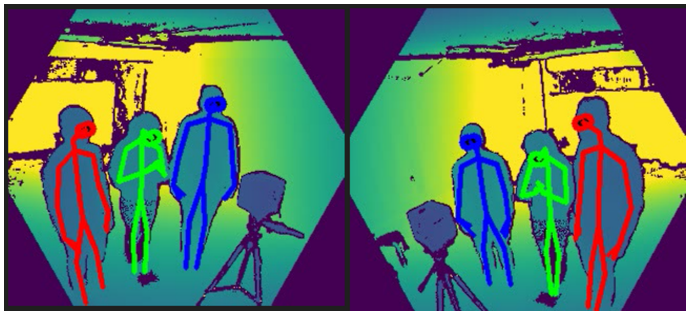


Figure 17: Three participants in the stand still period of manuscript 6.

The people counting software provided from the Kinect cameras were imprecise, because the cameras often lost the skeleton track of the subjects for a quick second or tracked the same skeleton as two different subjects. This resulted in incorrect labeling of the radar data in the form of quick changes in values and segments of less than a second. We created a script for correcting the label files. The script filters out the segments whose duration is less than 1 second, and closes the gap of the segment by setting the end of the previous segment to the start of the next segment. The script also limits the number of subjects to a maximum of the total number of subjects given by the manuscript. The script was run on all the label files, and the corrected label files were stored as NumPy files of shape $(N, 1)$ in the same folder as the corresponding RD maps.

5 Results and Discussion

In this section we will show the predictions on the test set, for each of the models, that gave the best performance on the validation set, we will then compare and discuss their performance and viability for solving the problem of people estimation.

For each model, many hyperparameters were tuned in terms of kernel size, number of filters and convolutional layers. Table 6 shows the best performing models on the validation set. The ResNet+LSTM model performed the worst in terms of every performance metric. After the ResNet+LSTM model, the CNN+GRU model exhibited better performance, followed by the ResNet, and finally the baseline CNN model, which achieved the highest overall performance among the four models. It is worth noting that the validation accuracy for the baseline CNN and ResNet was achieved on an imbalanced validation set as seen in Table 4. The validation accuracy for the CNN+GRU model was achieved on a validation set with a balanced number of recordings for each recording, but does not take into account that the frequency of the classes in recordings with two and three people, also consists of zero and one person. This means that the models might overfit towards predicting class zero and one during training.

Table 6: Best model performance on the validation set.

Model	Accuracy	Loss	Precision	Recall	F1-Score
Baseline CNN	**0.9258**	**0.2138**	**0.9297**	**0.9216**	**0.9256**
ResNet	0.9192	0.2392	0.9285	0.9101	0.9192
CNN+GRU	0.8870	0.3813	0.9000	0.8270	0.8620
ResNet+LSTM	0.8187	0.5127	0.8190	0.8187	0.8188

After having found the best performing model for each architecture, we used the test set for model predictions and compared the results. From the test set, we used one recording from each manuscript to ensure that the model was not overfitting on the validation set. Due to the lacking performance of the ResNet+LSTM model compared to the CNN+GRU, we did not include the former in the comparison.

From the model predictions we have summarized the best F1-score for each class in Table 7 and Table 8 where the former table's F1-scores are based on the recordings from manuscripts 1, 2, and 3, and the latter table's F1-scores are based on the recordings from manuscripts 4, 5, and 6. The decision to split them as such is due to the similarities in recordings for the two batches, giving a more representative comparison between the models and their performance.

Table 7: Best F1-score for each class based on manuscripts 1, 2, and 3.

Model	Best F1-score			
–	**0**	**1**	**2**	**3**
CNN+GRU	**1.0**	0.99	**0.93**	0.36
ResNet	**1.0**	**1.0**	0.42	0.60
Baseline CNN	**1.0**	0.99	0.75	**0.75**

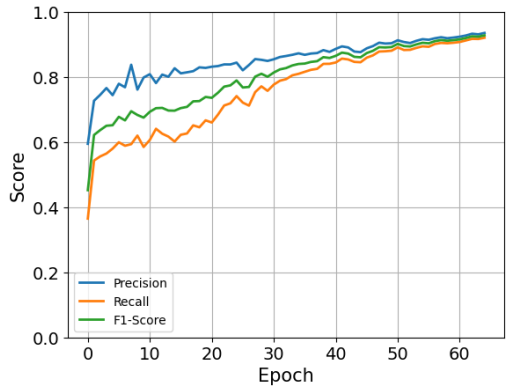
Table 8: Best F1-score for each class based on manuscripts 4, 5, and 6.

Model	Best F1-score			
–	**0**	**1**	**2**	**3**
CNN+GRU	**1.0**	0.94	0.24	0.00
ResNet	**1.0**	**1.0**	0.78	0.70
Baseline CNN	**1.0**	**1.0**	**0.96**	**0.95**

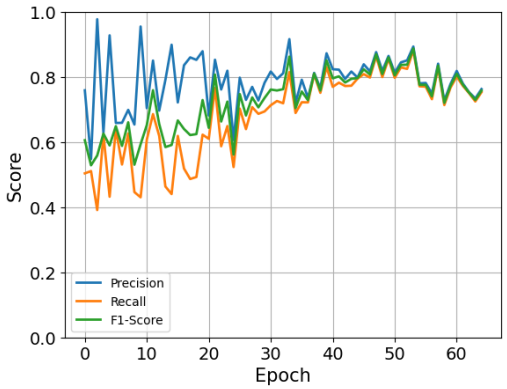
5.1 Predictions with CNN+GRU.

For the CNN+GRU models, we experimented with different frame models and temporal model hyperparameters. The frame model that worked best was the 2D CNN model with four convolutional layers where the filter size was $(1, 5)$, $(3, 3)$, $(1, 3)$, and $(3, 3)$ respectively. After every convolution, a normalization layer followed, and a max pooling layer was added after the last two normalization layers. The number of filters in the convolutional layers was 32, 64, 128, and 256 respectively. The activation function used was ReLU. The total number of learnable parameters is 5,129,220. For the recurrent layer, GRU was chosen as it outperformed the LSTM in terms of generalization and stability during training.

For the temporal model, the sequence length that was found to give the best training and validation accuracy was 50. We trained the model using both a sliding window and fixed-length sequences to evaluate their performances. For the sliding window approach, stride lengths ranging from 10 to 40 were tested. The sliding window approach was found to converge faster and achieve a higher accuracy than the fixed window approach on the training data, but was outperformed by the fixed-length sequences method on the validation data. This clearly indicates overfitting on the training data using the sliding window approach. Although the sliding window approach could capture temporal dynamics of the data more effectively, and could generate a larger amount of training data compared to fixed-length sequences, the introduction of redundant data in the sliding window approach could have caused the model to converge faster and to overfit on the training data. The fixed-length sequences approach, on the other hand, was found to take longer for the model to converge, but in return managed to generalize better on the validation data, and was therefore chosen as the temporal model for the CNN+GRU model. Figure 18 and 19 are plots of the performance metrics of a CNN+RNN model trained on fixed-length sequences and sliding window sequences respectively.

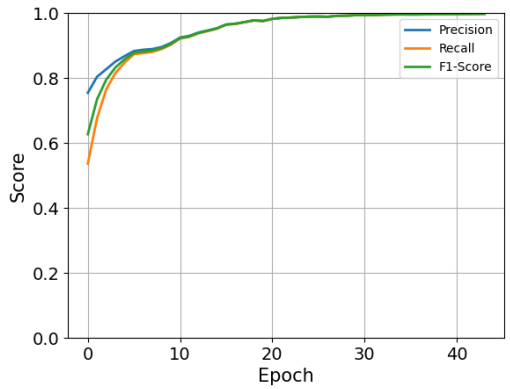


(a) Precision, recall and F1-score on the train set.

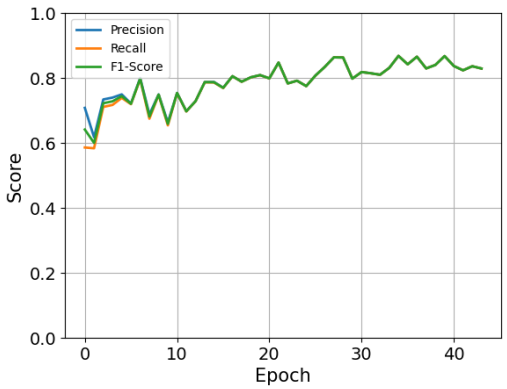


(b) Precision, recall and F1-score on the validation set.

Figure 18: Performance metrics for the CNN+GRU model trained on fixed-length sequences.



(a) Precision, recall and F1-score on the train set.



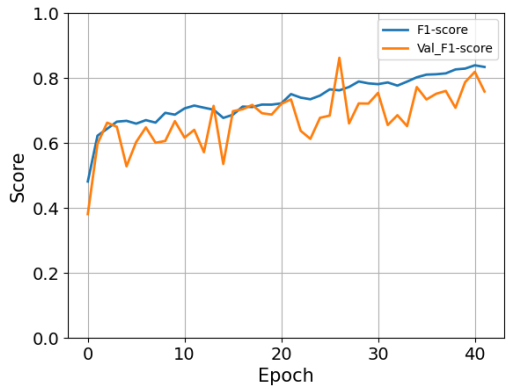
(b) Precision, recall and F1-score on the validation set.

Figure 19: Performance metrics for the CNN+GRU model trained on sliding window sequences.

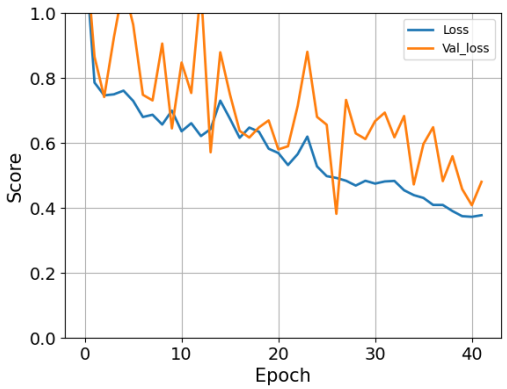
Converting the RD maps from power scale to the dB scale was found to improve the performance of the models. The reason for this could stem from the wide dynamic range of the power scale, making it difficult for ML models to effectively learn patterns and discriminate between different features. By converting the RD maps to the dB scale, the dynamic range is compressed, and variations in signal power become more discernible, allowing the model to capture more features. The dB scale also makes the identification of the noise floor easier, which the model in turn can learn to ignore. Further, the signal power becomes linearly related to the dB values, making them more responsive to standardization or normalization techniques.

For standardization or normalization, it was found that standardizing the RD maps resulted in better model performance. This could be due to the fact that the transformation of the data to have a mean of 0 and a standard deviation of 1 preserves the distribution characteristics of the RD maps, ensuring that the information contained in the relative differences between the pixel values is maintained. Normalization, on the other hand, transforms the data to a range between 0 and 1, which could result in the loss of information contained in the relative differences between the pixel values.

Figure 20 shows the results of training and validation F1-score and loss for the CNN+GRU model. The model was trained for 100 epochs with an early stopping to prevent overfitting, and a batch size of 64. From the plots, the early training epochs had significant spikes and drops in validation loss. This could be due to large weight updates given our learning rate. It seems that the model was training well, but a sudden drop in the validation loss occurred during training at epoch 26. The sharp drop in validation loss followed by a slower decrease and failure of the validation loss to reach its lowest point again could be the result of our learning rate scheduling. Due to the learning rate decay, it is possible that the learning rate decay caused slower convergence, preventing the model from reaching its previous low validation loss before the early stopping criterion was met.



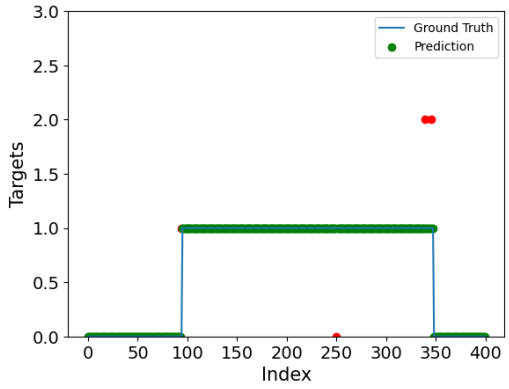
(a) Training and validation F1-score.



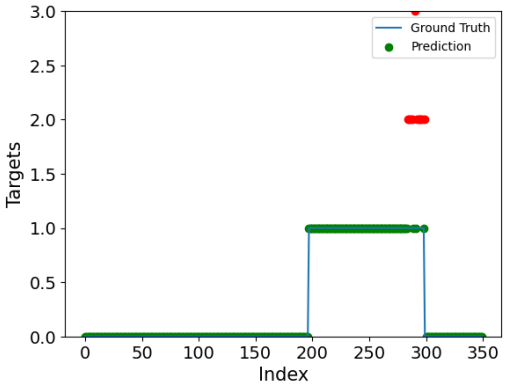
(b) Training and validation loss.

Figure 20: Performance metrics for the CNN+GRU model.

The results from predicting on the test set for one, two, and three targets have been plotted alongside the ground truth. The blue line is the ground truth, the red circles indicate wrong predictions, while the green circles indicate correct predictions. The results for the CNN+GRU model can be seen in Figures 21, 22, and 23.



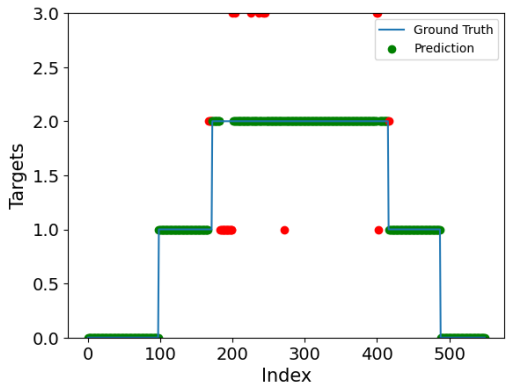
(a) Predictions on manuscript 1



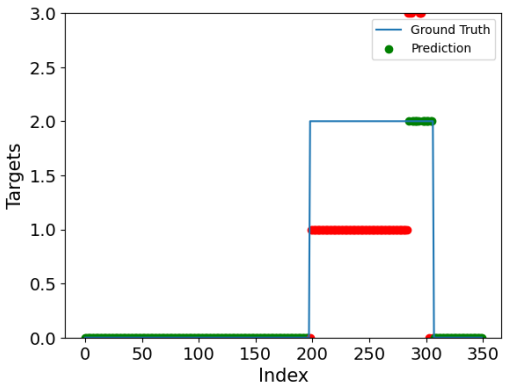
(b) Predictions on manuscript 4

Figure 21: CNN+GRU predictions on the test set for one target.

The model seems to predict correctly for one person but occasionally classifies incorrectly around when the person is either completely still or in movement. The classification of no target could be due to the model not being able to distinguish between the target and the noise floor when the target is standing completely still with movements only from breathing. The incorrect predictions of two and three targets happened close to the end of the recording, where the target is walking out of the scene, which in the RD maps could be hard to distinguish from other cases of similar movement patterns.



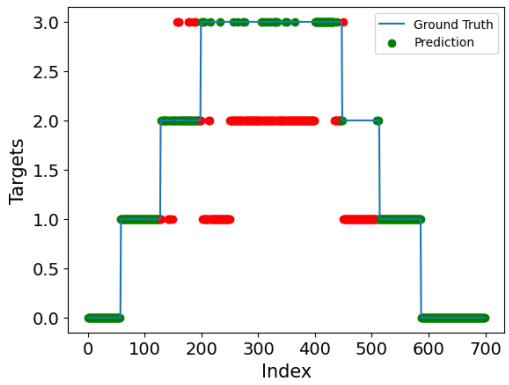
(a) Predictions on manuscript 2



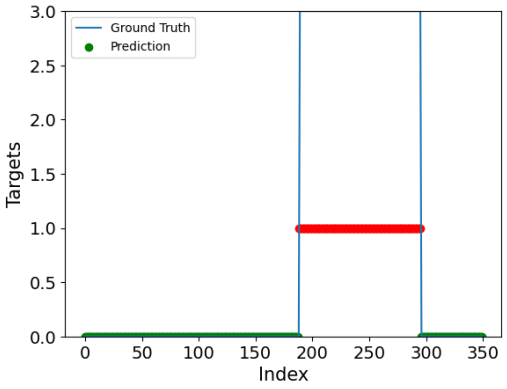
(b) Predictions on manuscript 5

Figure 22: CNN+GRU predictions on the test set for two targets.

The model shows promising performance in accurately predicting the number of people for manuscript 2 cases in Figure 22a. However, for the case of manuscript 5 in Figure 22b the model performed poorly, this could stem from the fact that the model was trained on more cases of manuscript 2 than manuscript 5. The model could also be overfitting on the manuscript 2 cases, which could explain the poor performance for the other. Manuscript 5 is also a challenging scenario for the model, as the targets are difficult to distinguish in the RD maps due to overlapping radar returns in both range and Doppler dimension. This ambiguity makes it a harder pattern to learn for the model, especially considering the limited amount of training samples.



(a) Predictions on manuscript 3



(b) Predictions on manuscript 6

Figure 23: CNN+GRU predictions on the test set for three targets.

Figure 23a and 22b show the predictions for three targets in manuscripts 3 and 6. The model performs poorly for manuscript 3, showing that the model have problems distinguishing multiple targets in the RD maps. The model performs even worse for manuscript 6, where it did not manage to predict any of the targets correctly. The model is likely overfitting on the training data, as the inherent nature of the recording process results in more samples of class 0 and 1 than class 2 and 3.

5.2 Predictions with ResNet

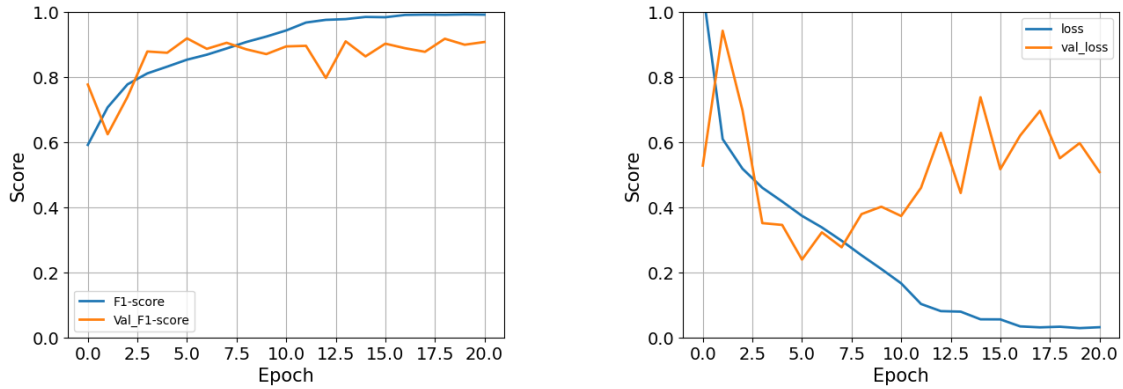
The ResNet starts with an initial convolutional layer that applies a 5x5 kernel to the input maps. Batch normalization and activation functions (ReLU) are then applied to enhance the model's ability to learn important features. Max pooling is utilized to reduce the spatial dimensions while preserving essential information.

The core of the model consists of residual blocks, which are responsible for learning deep representations. These blocks enable the model to capture intricate patterns and features by utilizing shortcut connections and skip connections. The residual blocks are stacked in a hierarchical manner, with increasing filter sizes (64, 128, 256, 512) to extract increasingly complex features.

Global average pooling is applied to obtain a condensed representation of the features learned by the previous layers. This reduces the spatial dimensions while preserving the important information required for classification. A fully connected dense layer with 256 units and ReLU activation is added to further capture high-level representations.

To mitigate overfitting, dropout regularization is applied, randomly disabling a fraction of neurons during training. This promotes generalization and prevents the model from relying too heavily on specific features. Finally, a dense layer with softmax activation is used as the output layer to produce class probabilities.

The model is trained using the Adam optimizer with a learning rate of 0.001. The loss function employed is categorical cross-entropy, suitable for multiclass classification tasks. Additionally, precision, recall, and accuracy metrics are utilized to evaluate the model's performance. The total number of learnable parameters in the model is 12,494,532.

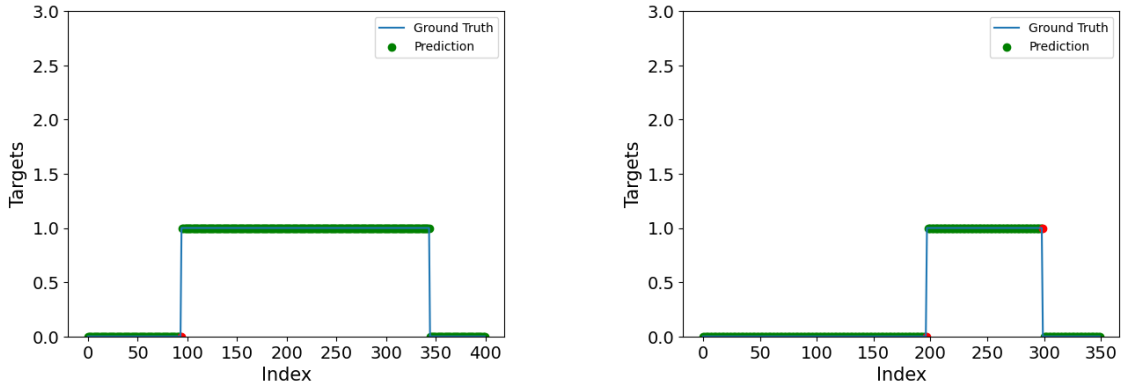


(a) Training and validation F1-score.

(b) Training and validation loss.

Figure 24: Performance metrics for the ResNet model.

From Figure 24a and 24b, we can see that the model starts to overfit after only five epochs. This is likely due to the limited amount of training data and the deep architecture of the model, indicating that a simpler model architecture would be more suitable, or that more training data is needed.

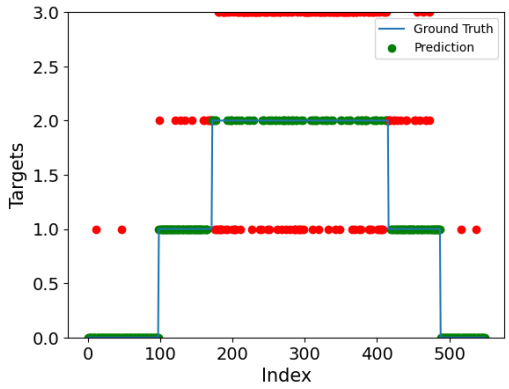


(a) Predictions on manuscript 1

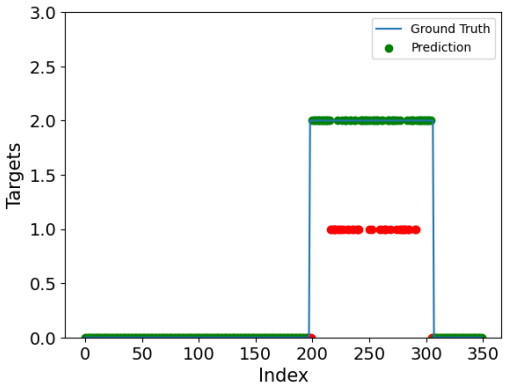
(b) Predictions on manuscript 4

Figure 25: ResNet predictions on the test set for one target.

As expected, the model performs well on the test set for one target, as seen in Figure 25a and 25b.

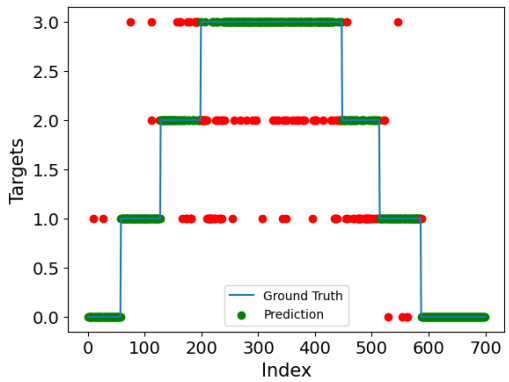


(a) Predictions on manuscript 2

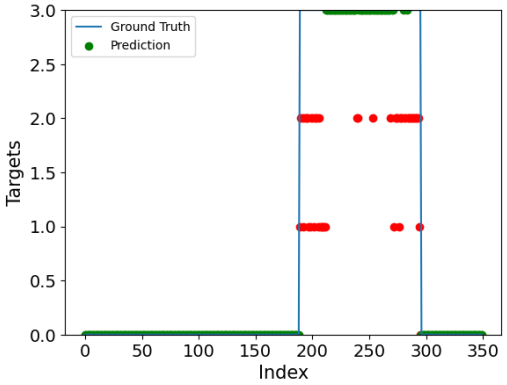


(b) Predictions on manuscript 5

Figure 26: ResNet predictions on the test set for two targets.



(a) Predictions on manuscript 3



(b) Predictions on manuscript 6

Figure 27: ResNet predictions on the test set for three targets.

From the plots of the predictions Figures 26 and 27, we can observe that the model is having issues when more than one person is present in the scene. The model predicts better for manuscripts 5 and 6 than for manuscripts 2 and 3, but is still not consistent in its predictions. The complex architecture might have caused the model to overfit on the training data.

5.3 Predictions with CNN

The model starts with an input layer that takes in data of shape $(16, 128, 3)$. It then applies several convolutional layers with different filter sizes and kernel sizes, each followed by a ReLU activation function. The first convolutional layer has 16 filters with a kernel size of $(1, 1)$, the second has 32 filters with a kernel size of $(1, 3)$, the third has 64 filters with a kernel size of $(3, 3)$, the fourth has 128 filters with a kernel size of $(3, 3)$, and the fifth has 256 filters with a kernel size of $(3, 3)$.

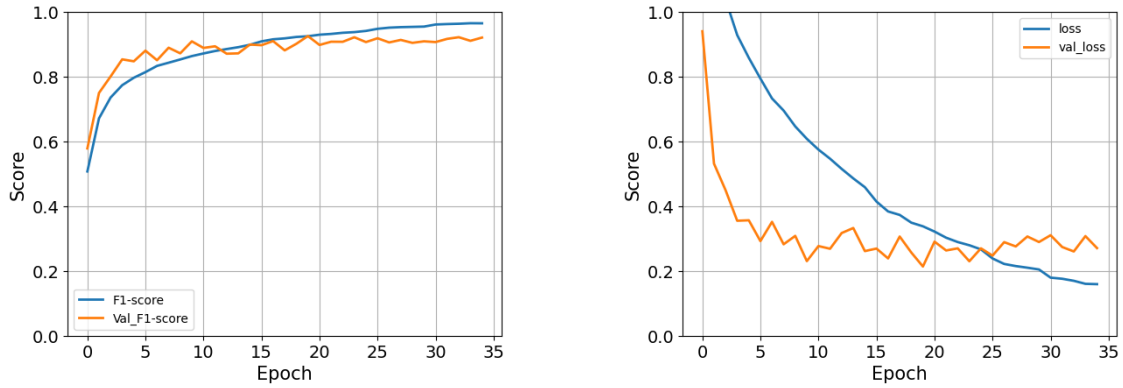
Max pooling layers are applied after some of the convolutional layers to reduce the spatial dimensions of the feature maps. The model also includes dropout layers to help prevent overfitting. After the final convolutional layer, a global average pooling layer is applied to further reduce the spatial dimensions.

Next, the model includes a dense layer with 128 units and a ReLU activation function. A dropout layer is added with a dropout rate of 0.5 to further regularize the model. The output layer

consists of a dense layer with num_classes units and a softmax activation function for multiclass classification.

The model is compiled with the Adam optimizer using a learning rate of 0.001. The loss function is categorical cross-entropy, and the metrics used for evaluation are precision, recall, and accuracy. The total number of learnable parameters is 422,564.

For the preprocessing of the RD maps, the same methods as described in Section 3.2.3 were used. It was found that the range compensation method did not improve the performance of the model, and was therefore not used. The model was trained for 100 epochs with a batch size of 64, and the model with the lowest validation loss and highest validation accuracy was saved. The model was trained on an NVIDIA A100 Tensor Core GPU provided by Google Colab with an early stopping of 15 epochs to stop the model from overfitting. The results from the training can be seen in Figure 28.

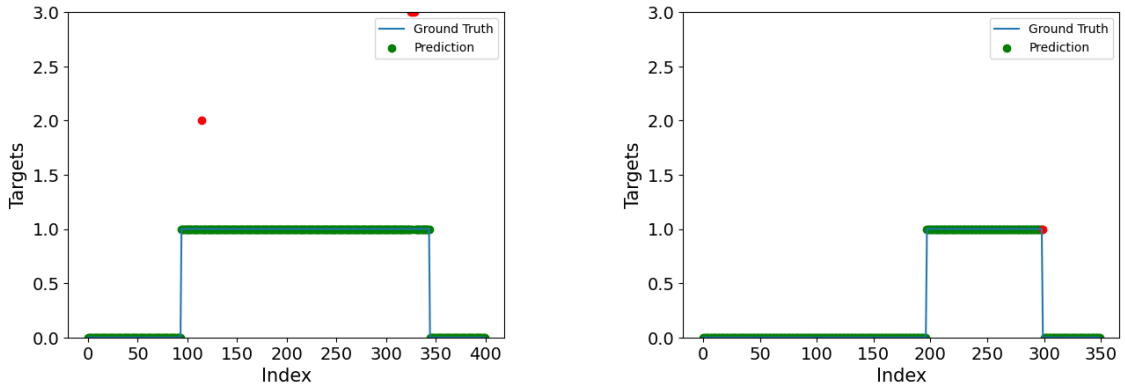


(a) Training and validation F1-score.

(b) Training and validation loss.

Figure 28: Performance metrics for the baseline CNN model.

Below, we have plotted the results from predicting on the test set for one, two, and three targets alongside the ground truth.

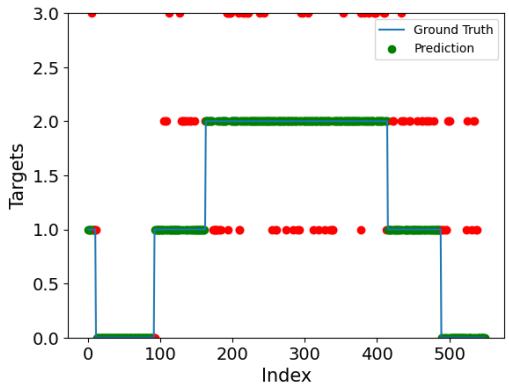


(a) Predictions on manuscript 1.

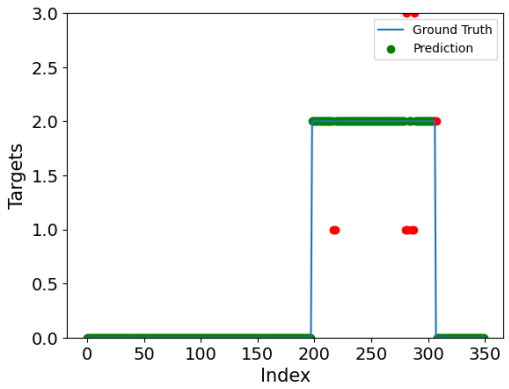
(b) Predictions on manuscript 4.

Figure 29: Baseline CNN predictions on the test set for one target.

Just like the previous two models, the classification task for one person is satisfactory and proves that the model manages to reject clutter from multipath, objects, and noise.



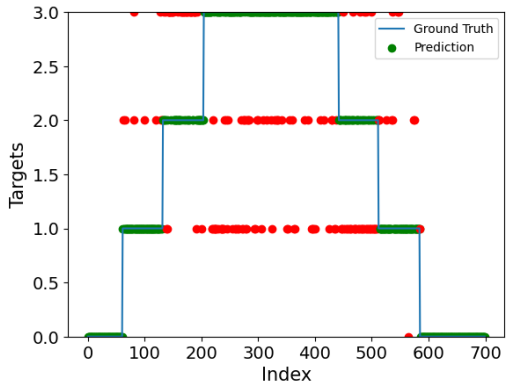
(a) Predictions on manuscript 2.



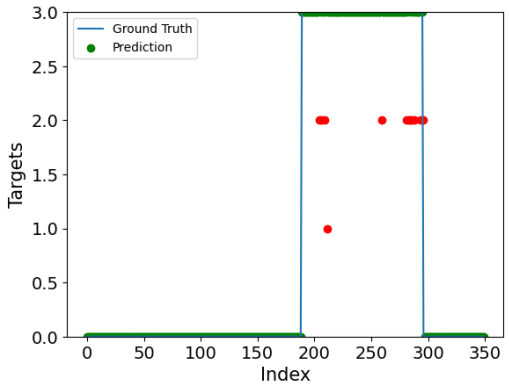
(b) Predictions on manuscript 5.

Figure 30: Baseline CNN predictions on the test set for two targets.

Moving on to two targets, we can observe that the model is having issues with predicting consistently for manuscript 2, but predicts incredibly well for manuscript 5. Manuscript 5 should theoretically be the more challenging case as the two targets walk synchronously at the same angle towards the radar. The RD map from such a scenario would only show as one target, but the model manages to distinguish between the two targets and predict correctly.



(a) Predictions on manuscript 3.



(b) Predictions on manuscript 6.

Figure 31: Baseline CNN predictions on the test set for three targets.

Finally, for three targets, the model suffers from the same issues as for two targets, where it has difficulties predicting correctly for manuscript 3, but predicts correctly for manuscript 6.

One possible explanation for the superior performance of the simple CNN model could be its ability to effectively capture and extract relevant features from the RD maps. The simple CNN model, with its series of convolutional layers followed by pooling and dense layers, demonstrated strong feature extraction capabilities. The convolutional layers with small filter sizes were able to capture local patterns and structures within the RD maps. The subsequent pooling layers helped to reduce the spatial dimensions and retain the most salient features. These extracted features were then processed by the dense layers to make accurate predictions. The simplicity of the CNN architecture might have enabled better generalization and avoidance of overfitting on the given dataset.

On the other hand, the ResNet and CNN+GRU models, despite their more complex architectures and the inclusion of recurrent layers, did not exhibit the same level of performance as the simple CNN model. The ResNet model, known for its deep residual connections and skip connections, was expected to capture more complex patterns and improve the flow of gradients during training. However, the deeper architecture might have introduced additional complexity and increased the risk of overfitting, leading to poorer generalization performance. Furthermore, a pretrained ResNet

would have been ideal to use as they already are trained for object detection, but due to the shape of the RD maps, it was not possible to implement. The CNN+GRU model, which combines CNN and GRU layers, aimed to leverage both spatial and temporal information in the RD maps. However, the model's ability to effectively capture the temporal dependencies might have been compromised by the limited temporal context provided by the relatively small number of frames used.

It is also worth considering the nature of the dataset and the characteristics of the RD maps. The simple CNN model, with its straightforward architecture, might have been better suited for this particular dataset. The RD maps, representing the radar reflections from people moving within a scene, are inherently spatial in nature. Therefore, a model that excels at spatial feature extraction, such as the simple CNN model, is likely to perform well. The CNN+GRU models, while incorporating recurrent layers to capture temporal dependencies, might not have fully exploited the spatial characteristics of the data, while the ResNet model, with its deep architecture, might have been too complex for the dataset.

One notable limitation of our study is the scarcity of training data available for the multiclass classification task of RD maps for people counting. The success of deep learning models, such as CNNs and recurrent models, heavily relies on a large and diverse dataset to learn representative features and patterns. However, due to the fact that the data collection started during the project study, the size of our dataset was limited.

The limited training data can have a significant impact on the model's performance and generalization ability. Insufficient data can lead to overfitting, where the model memorizes the training examples without truly understanding the underlying patterns. As a result, the model may struggle to generalize well to unseen data, leading to suboptimal performance on the validation or test sets.

In our case, the scarcity of training data might have affected the performance of the more complex models, such as ResNet and CNN+GRU. These models typically have a higher number of learnable parameters and require a larger amount of data to effectively optimize and generalize. With a limited dataset, these models might not have had enough diverse examples to capture the complex variations and relationships present in the RD maps.

To mitigate the impact of the limited training data, we employed techniques such as data augmentation, which involved generating additional synthetic examples by applying random transformations and perturbations to the existing data. This helped to artificially increase the size and diversity of the training set and encouraged the model to generalize better. However, the augmentation process might not fully capture the true variability of the data, and its effectiveness might be limited.

The height placement of a radar system can play a significant role in the characteristics of the radar return signals, particularly in scenarios involving NLOS conditions. In our study, the radar was placed at a height of 1.5 meters, which is relatively low compared to the typical height of a person. This could have affected the characteristics of the radar return signals, and consequently the performance of the models. Higher radar placements may provide better line-of-sight visibility and reduced signal blockage, resulting in stronger and more reliable radar returns. On the other hand, lower radar placements may increase the likelihood of NLOS conditions, leading to weaker and more distorted signals. Therefore, it would be interesting to explore the impact of radar height on the performance of the models in future studies.

For the preprocessing, minimal steps were taken to remove noise and clutter from the RD maps before training. Noise and clutter, arising from sources such as instrumentation imperfections or environmental factors, can introduce unwanted artifacts and distortions in the RD maps. However, in some cases, there may be benefits to retaining noise and clutter in the training process. Noise, despite being typically considered as unwanted interference, can contain complex patterns that may carry valuable information. By allowing the model to learn and adapt to the noise patterns, it may be able to better distinguish between noise and the target signals. Clutter, which refers to unwanted reflections from objects in the scene, can also contain useful information.

The results our model have achieved are promising and compares well against the state-of-the-art methods, where only the handcrafted feature and novel ML architecture proposed by Choi et al.

[17] outperformed our model.

NOVELDA Ultra-low Power Presence Sensor opens up the possibility to implement radar systems that provides both range and azimuth angle resolution with an even lower power consumption than the current radar systems utilizing a single antenna. The spatial information provided by the dual-antenna configuration allows for better localization and differentiation of multiple targets within the detection zone. By utilizing beamforming to steer the radar beam, the radar can discern, to an extent, the direction from which the targets approach. This additional spatial information not only enhances the accuracy of target localization but also provides crucial insights into target dynamics, leading to more precise and reliable people counting results. Moreover, the dual-antenna's ability to resolve multiple targets with distinct azimuth angles contributes to mitigating the issue of target occlusions and overlapping trajectories, which is often encountered in crowded environments.

6 Conclusion

This study explored the use of range-Doppler (RD) maps from the NOVELDA Ultra-low Power Presencec Sensor as features for machine learning (ML) classification of how many people there are in a 4×4m grid. We have created a People Counting (PC) dataset containing RD maps, of three beams pointing in 0°, 20°, and -20° azimuth angle, of one to three targets in a 4×4m grid for people counting. Two versions of the dataset were created: one for training on individual frames of RD maps, and one for training on sequential data. We created a GUI for manually labeling the recordings, which were based on six different manuscripts describing the activity performed. The manuscript can be found in Appendix A. The dataset is not publicly available. Further, a script was developed during the study to automatically label the recordings, which saved significant time and resources. The RD maps were extracted from the radar sensor, where each cell in the RD map represents the absolute target return signal power. The sensor was placed 1.50 meters above the ground towards the grid making classifications more difficult as the radar is more prone to NLOS targets.

The high dimensions of the RD maps, and the high sample frequency from the radar with regards to sequential models, presented a problem due to memory issues during training. We decimated the range bins by a factor of four, removed the excess range bins that were further away than four meters, and applied a 128 point FFT such that each RD map was reduced to a size of 16×128 matrix. Further, for the sequential dataset we downsampled the recordings to four frames per second (FPS). The high dynamic range of the power values in the RD maps were compressed using a dB conversion, and the values were standardized using the mean and standard deviation of the training set. It was found that the dB conversion and standardization stabilized the training and improved the performance of the models. Range compensation was applied, but it was found that it neither improved nor significantly reduced the performance of the models. The two final datasets are presented in Table 5 and Table 4.

We investigated the performance of three different models, namely a CNN model, ResNet, and CNN+GRU, for the task of multiclass classification of three-channel RD maps, each channel representing angles in 0°, 20°, and -20° azimuth, for people counting. The final shape for each time step was (16, 128, 3). Surprisingly, we observed that the simple CNN model outperformed both the ResNet and CNN+GRU models in terms of overall performance as presented in Table 6. The simple architecture of the CNN might have enabled better generalization and avoidance of overfitting on the dataset, while the other two models were much more complex in architecture. The deep residual connections and inclusion of recurrent layers for the ResNet and CNN+GRU model might have introduced additional complexity, leading to an increased risk of overfitting on a sparse training set and a poorer generalization to unseen data.

Despite the constraint of limited training data in our study, we believe our findings provide valuable insights into the performance of different models for the multi-class classification of RD maps for people counting in difficult scenarios. The results of our experiments suggest that the three RD maps from the radar sensor can be used as features for the people counting task. The results also suggest that even with a challenging setup where the radar mount is placed at a height of 1.50 meters, making the radar more prone to NLOS targets, the radar data is still useful. The CNN performed very well for the challenging classification cases of manuscripts 5 and 6. It would be almost impossible to resolve such targets for a single-antenna radar because the targets would be presented as a single peak in the RD map. Future studies with larger and more diverse datasets would be necessary to validate and extend our findings, ultimately leading to more reliable and robust models in real-world scenarios.

6.1 Future Work

In future studies, it is essential to address the issue of limited training data by exploring strategies to collect or generate more labeled examples. This could involve acquiring additional RD maps through extended data collection efforts or leveraging techniques such as semi-supervised learning or transfer learning to make the most of available data. Increasing the size and diversity

of the dataset would enable the models to learn more robust representations and improve their performance and generalization capabilities.

It is important to note that the need for more training data is a common challenge in many real-world machine learning applications, particularly in fields where data collection is time-consuming, costly, or subject to limitations. Addressing this limitation requires careful consideration of data collection strategies, and exploring alternative approaches to make the most of the available data.

We focused on utilizing RD maps as the primary input for training our models for the task of multi-class classification of people counting. However, it is worth considering the potential benefits and challenges of using other types of input data, such as raw signals from the radar. The use of raw signals would enable the models to learn more complex representations and potentially improve their performance. However, it would also require more complex preprocessing and feature extraction steps, which could be challenging to implement and optimize. Promising results have already been achieved using raw signals from Novelda's X4 sensor with a single antenna [17], and it would be interesting to explore the use of raw signals from their dual-antenna IR-UWB radar in future studies.

References

[1] S. Parameswari and C. Chitra, 'Compact textile uwb antenna with hexagonal for biomedical communication', *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–8, 2021.

[2] J. Bourqui and E. C. Fear, 'Shielded uwb sensor for biomedical applications', *IEEE Antennas and Wireless Propagation Letters*, vol. 11, pp. 1614–1617, 2012.

[3] R. Liu, C. Yuen, T.-N. Do, D. Jiao, X. Liu and U.-X. Tan, 'Cooperative relative positioning of mobile users by fusing imu inertial and uwb ranging information', in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 5623–5629.

[4] G. Tiberi and M. Ghavami, 'Ultra-wideband (UWB) systems in biomedical sensing', *Sensors*, vol. 22, no. 12, p. 4403, Jun. 2022. DOI: 10.3390/s22124403. [Online]. Available: https://doi.org/10.3390/s22124403.

[5] M. Husaini, L. M. Kamarudin, A. Zakaria *et al.*, 'Non-contact breathing monitoring using sleep breathing detection algorithm (SBDA) based on UWB radar sensors', *Sensors*, vol. 22, no. 14, p. 5249, Jul. 2022. DOI: 10.3390/s22145249. [Online]. Available: https://doi.org/10.3390/s22145249.

[6] R. de Goederen, S. Pu, M. S. Viu *et al.*, 'Radar-based sleep stage classification in children undergoing polysomnography: A pilot-study', *Sleep Medicine*, vol. 82, pp. 1–8, Jun. 2021. DOI: 10.1016/j.sleep.2021.03.022. [Online]. Available: https://doi.org/10.1016/j.sleep.2021.03.022.

[7] J. Choi, D. Yim and S. Cho, 'People counting based on ir-uwb radar sensor', *IEEE Sensors Journal*, vol. 17, no. 17, pp. 5717–5727, Sep. 2017.

[8] *Location Detection and Tracking of Moving Targets by a 2D IR-UWB Radar System — doi.org*, https://doi.org/10.3390/s150306740, [Accessed 21-Jul-2023].

[9] S. Chang, N. Mitsumoto and J. Burdick, 'An algorithm for uwb radar-based human detection', Jun. 2009, pp. 1–6. DOI: 10.1109/RADAR.2009.4976999.

[10] S. Chang, M. Wolf and J. W. Burdick, 'Human detection and tracking via ultra-wideband (uwb) radar', in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 452–457. DOI: 10.1109/ROBOT.2010.5509451.

[11] *NOVELDA – The heart of our innovation is the unique UWB radar System on Chip — novelda.com*, https://novelda.com/technology/datasheets, [Accessed 21-Jul-2023].

[12] A. Santra and S. Hazra, *Deep Learning Applications, Volume 2*. Artech House Publishers, 2021.

[13] J.-H. Choi, J.-E. Kim and K.-T. Kim, 'People counting using ir-uwb radar sensor in a wide area', *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5806–5821, 2020.

[14] J.-H. Choi, J.-E. Kim, K.-T. Kim *et al.*, 'Learning-based people counting system using an ir-uwb radar sensor', *The Journal of Korean Institute of Electromagnetic Engineering and Science*, vol. 30, no. 1, pp. 28–37, 2019.

[15] J.-E. Kim, J.-H. Choi and K.-T. Kim, 'Robust detection of presence of individuals in an indoor environment using ir-uwb radar', *IEEE Access*, vol. 8, pp. 108 133–108 147, 2020.

[16] J.-H. Choi, J.-E. Kim, N.-H. Jeong, K.-T. Kim and S.-H. Jin, 'Accurate people counting based on radar: Deep learning approach', in *2020 IEEE Radar Conference (RadarConf20)*, 2020, pp. 1–5. DOI: 10.1109/RadarConf2043947.2020.9266496.

[17] J.-H. Choi, J.-E. Kim and K.-T. Kim, 'Deep learning approach for radar-based people counting', *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, Sep. 2021. DOI: 10.1109/JIOT.2021.3113671.

[18] R. Bao and Z. Yang, 'Cnn-based regional people counting algorithm exploiting multi-scale range-time maps with an ir-uwb radar', *IEEE Sensors Journal*, vol. 21, no. 12, pp. 13 704–13 713, 2021. DOI: 10.1109/JSEN.2021.3071941.

[19] M. Stephan, S. Hazra, A. Santra, R. Weigel and G. Fischer, 'People counting solution using an fmcw radar with knowledge distillation from camera data', in *2021 IEEE Sensors*, IEEE, 2021, pp. 1–4.

[20] M. Richards, J. Scheer and W. Holm, *Principles of Modern Radar: Basic Principles*. SciTech Publishing, 2010.

[21] D. Winter, 'Human balance and posture control during standing and walking', *Gait and Posture*, vol. 3, no. 4, pp. 193–214, Dec. 1995.

[22] T. Malone, D. Rus and R. Laubacher, 'Artificial intelligence and the future of work', Massachusetts Institute of Technology, Tech. Rep., 2020.

[23] F. Rosenblatt, 'The perceptron: A probabilistic model for information storage and organization in the brain', *Psychological Review*, vol. 65, no. 6, pp. 386–408, Nov. 1958.

[24] Mayranna, *Perceptron*, From Wikimedia Community, the free media repository, 2013.

[25] *FRANCKE PEIXOTO — analyticsvidhya.com*, https://www.analyticsvidhya.com/blog/author/franckepeixoto/, [Accessed 21-Jul-2023].

[26] S. Yang, X. Yu and Y. Zhou, 'Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example', Jun. 2020, pp. 98–101. DOI: 10.1109/IWECAI50956.2020.00027.

[27] *NOVELDA Introduces New Ultra-Low Power UWB Sensor — novelda.com*, https://novelda.com/news/ultra-low-power-presence-sensor, [Accessed 21-Jul-2023].

[28] *Novelda UWB Ultra-low power sensor — novelda.com*, https://novelda.com/ultra-low-power-sensor, [Accessed 21-Jul-2023].

A Recording manuscripts

Manuscript 1: Chessboard one person. The recording starts with 30 seconds of no target in the detection zone, silence. After the silence period, one participant walks into the detection zone from one of the four squares furthest away from the radar. The participant walks in a straight line towards a random square in the grid, and either stands still, or sits (upright with minimal movement or relaxed) in a preplaced chair facing towards or away from the sensor. The participant stays in the square for one minute. After the one minute period, the participant walks out of the detection zone exiting from the same square as they entered.

Manuscript 2: Chessboard two person. The recording starts with 30 seconds of no target in the detection zone, silence. After the silence period, the first participant walks into the detection zone from one of the four squares furthest away from the radar. The participant walks in a straight line towards a random square in the grid, and either stands still, or sits (upright with minimal movement or relaxed) in a preplaced chair facing towards or away from the sensor. The first participant stays in the square for one minute. 10 seconds after the first participant enters, the second participant walks into the detection zone from one of the four squares furthest away from the radar. The second participant walks in a straight line towards a random square which is not occupied in the grid, and either stands still, or sits (upright with minimal movement or relaxed) in a preplaced chair facing towards or away from the sensor. The second participant stays in the square for one minute. After the first one minute period, the first participant walks out of the detection zone exiting from the same square as they entered. After the second one minute period, the second participant walks out of the detection zone exiting from the same square as they entered.

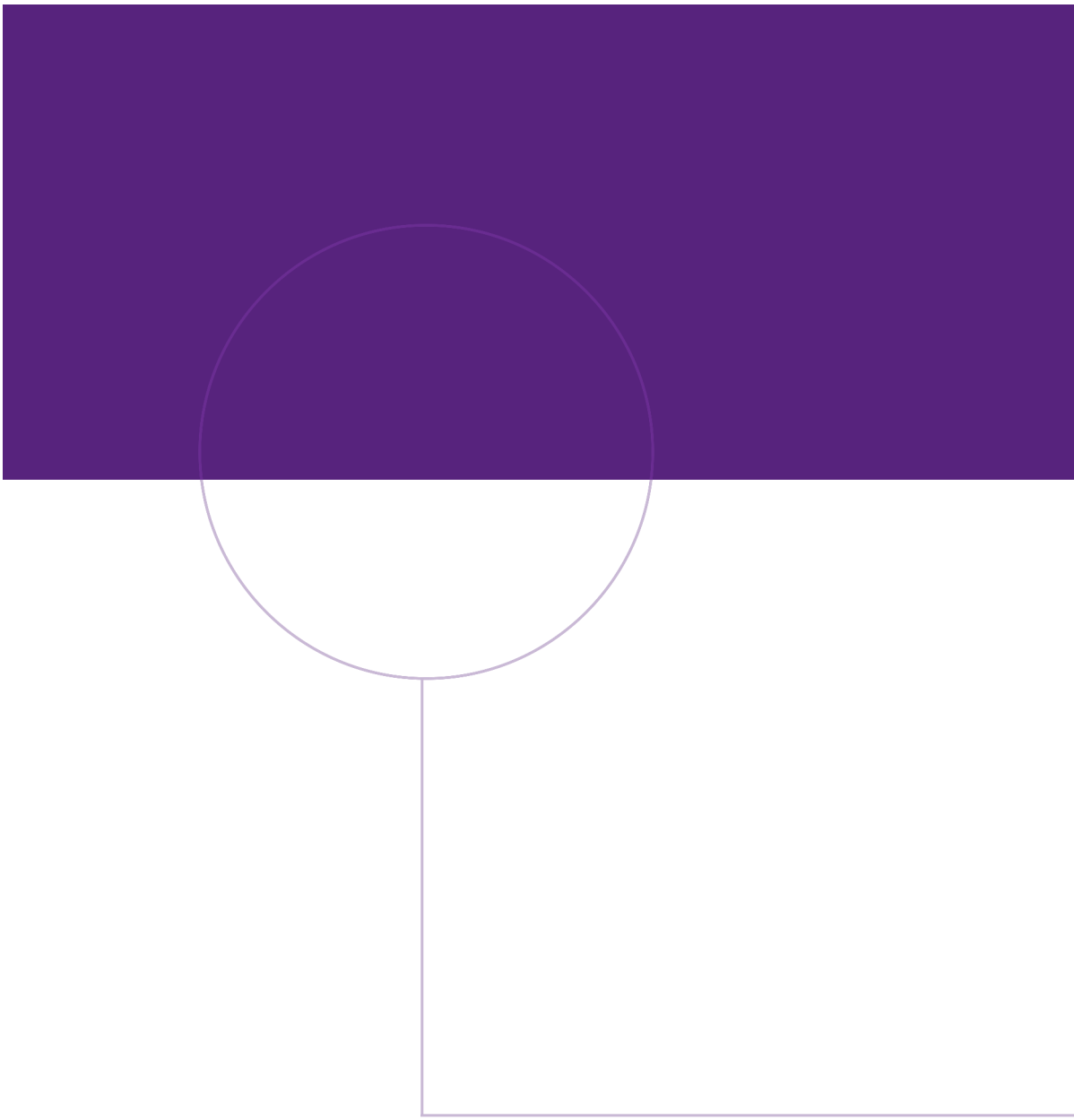
Manuscript 3: Chessboard three person. The recording starts with 30 seconds of no target in the detection zone, silence. After the silence period, the first participant walks into the detection zone from one of the four squares furthest away from the radar. The participant walks in a straight line towards a random square in the grid, and either stands still, or sits (upright with minimal movement or relaxed) in a preplaced chair facing towards or away from the sensor. The first participant stays in the square for one minute. 10 seconds after the first participant enters, the second participant walks into the detection zone from one of the four squares furthest away from the radar. The second participant walks in a straight line towards a random square which is not occupied in the grid, and either stands still, or sits (upright with minimal movement or relaxed) in a preplaced chair facing towards or away from the sensor. The second participant stays in the square for one minute. 10 seconds after the second participant enters, the third participant walks into the detection zone from one of the four squares furthest away from the radar. The third participant walks in a straight line towards a random square which is not occupied in the grid, and either stands still, or sits (upright with minimal movement or relaxed) in a preplaced chair facing towards or away from the sensor. The third participant stays in the square for one minute. After the first one minute period, the first participant walks out of the detection zone exiting from the same square as they entered. After the second one minute period, the second participant walks out of the detection zone exiting from the same square as they entered. After the third one minute period, the third participant walks out of the detection zone exiting from the same square as they entered.

Manuscript 4: Walk in angle one person. The recording starts with 30 seconds of no target in the detection zone, silence. After the silence period, one participant walks in along one of the tape lines (-30°, -20°, -10°, 0°, +10°, +20°, +30° seen from the radar) from five meters to one meters within 10 seconds. The participant stands still for 15 seconds. After the 15 second period, the participant walks out of the detection zone along the same tape line as they entered within 10 seconds.

Manuscript 5: Walk in angles two person, synchronized. The recording starts with 30 seconds of no target in the detection zone, silence. After the silence period, two participants walks

in simultaneously along one of the tape lines in mirrored angles (±30°, ±20°, ±10° seen from the radar) from five meters to one meters within 10 seconds. The participants stands still for 15 seconds. After the 15 second period, the participants walks out of the detection zone along the same tape line as they entered within 10 seconds.

Manuscript 6: Walk in angles three person, synchronized. The recording starts with 30 seconds of no target in the detection zone, silence. After the silence period, two participants walks in simultaneously along one of the tape lines in mirrored angles (±30°, ±20°, ±10° seen from the radar) from five meters to one meters within 10 seconds. The third participant walks in simultaneously with the other two participants along the middle tape line (0° seen from the radar) from five meters to one meters within 10 seconds. The participants stands still for 15 seconds. After the 15 second period, the participants walks out of the detection zone along the same tape line as they entered within 10 seconds.



NTNU
Norwegian University of
Science and Technology

NOVELDA