HUMAN ACTIVITY CLASSIFICATION WITH DEEP LEARNING USING FMCW RADAR

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

By
Mert Ege
September 2022

Human Activity Classification with Deep Learning using FMCW Radar
By Mert Ege
September 2022

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Ömer Morgül(Advisor)

Tolga Çukur

Mehmet Önder Efe

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

HUMAN ACTIVITY CLASSIFICATION WITH DEEP LEARNING USING FMCW RADAR

Mert Ege
M.S. in Electrical and Electronics Engineering
Advisor: Ömer Morgül
September 2022

Human Activity Recognition (HAR) has recently attracted academic research attention and is used for purposes such as healthcare systems, surveillance-based security, sports activities, and entertainment. Deep Learning is also frequently used in Human Activity Recognition, as it shows superior performance in subjects such as Computer Vision and Natural Language Processing.

FMCW radar data is a good choice for Human Activity Recognition as it works better than cameras under challenging situations such as rainy and foggy conditions. However, the work in this field does not progress as dynamically as in the camera-based area. This can be attributed to radar-based models that do not perform as well as camera-based models.

This thesis proposes four new models to improve HAR performance using FMCW radar data. These models are CNN-based, LSTM-based, LSTM- and GRU-based, and Siamese-based. For feature extraction, the CNN-based model uses CNN blocks, the LSTM-based model uses LSTM blocks, and the LSTM- and GRU-based model uses LSTM and GRU blocks in parallel. Furthermore, the Siamese-based model is fed in parallel from three different radars (multi-input). Due to the Siamese Network nature, parallel paths will have the same weight. On the other hand, after feature extraction, all models use dense layers to classify human motion.

To our best knowledge, the Siamese-based model is used for the first time in multi-input data for the classification of human movement. This model outperforms the state-of-the-art models by using various features of radars operating at different frequencies in terms of classification accuracy. All codes and their

results can be found at "https://github.com/mertege/Thesis_Experiments".

Keywords: Human Activity Classification, FMCW Radar, Deep Learning, Siamese Network, micro-Doppler, Data Augmentation.

ÖZET

FMCW RADAR DATASINI KULLANARAK DERİN ÖĞRENME İLE İNSAN ETKİNLİĞİ SINIFLANDIRMASI

Mert Ege
Elektrik ve Elektronik Mühendisliği, Yüksek Lisans
Tez Danışmanı: Ömer Morgül
Eylül 2022

İnsan Aktivitesi Tanıma (HAR) son zamanlarda akademik araştırmaların ilgisini çekmiştir ve sağlık sistemleri, gözetime dayalı güvenlik, spor aktiviteleri ve eğlence gibi amaçlar için kullanılmaktadır. Derin Öğrenme, Bilgisayarla Görme ve Doğal Dil İşleme gibi konularda üstün performans gösterdiği için İnsan Aktivitesi Tanıma'da da sıklıkla kullanılmaktadır.

FMCW radar verileri, yağmurlu ve sisli koşullar gibi zorlu durumlarda kameralardan daha iyi çalıştığı için İnsan Aktivitesi Tanıma için iyi bir seçimdir. Ancak bu alandaki çalışmalar kamera tabanlı alandaki kadar dinamik bir şekilde ilerlememektedir. Bu, kamera tabanlı modeller kadar iyi performans göstermeyen radar tabanlı modellere bağlanabilir.

Bu tez, FMCW radar verilerini kullanarak HAR performansını iyileştirmek için dört yeni model önermektedir. Bu modeller CNN tabanlı, LSTM tabanlı, LSTM ve GRU tabanlı ve Siyam tabanlıdır. Özellik çıkarımı için CNN tabanlı model CNN bloklarını kullanır, LSTM tabanlı model LSTM bloklarını kullanır ve LSTM ve GRU tabanlı model paralel olarak LSTM ve GRU bloklarını kullanır. Ayrıca, Siyam tabanlı model, üç farklı radardan paralel olarak beslenir (çoklu giriş). Siyam Ağının doğası gereği paralel yollar aynı ağırlığa sahip olacaktır. Öte yandan, özellik çıkarımından sonra, tüm modeller insan hareketini sınıflandırmak için yoğun katmanlar kullanır.

Bildiğimiz kadarıyla, Siyam tabanlı model, insan hareketinin sınıflandırılması için çok girdili verilerde ilk kez kullanılıyor. Bu model, sınıflandırma doğruluğu açısından farklı frekanslarda çalışan radarların çeşitli özelliklerini kullanarak en

son modellerden daha iyi performans göstermektedir.

Tüm kodlar ve sonuçları "https://github.com/mertege/Thesis_Experiments" adresinde bulunabilir.

Anahtar sözcükler: İnsan Aktivite Sınıflandırması, FMCW Radar, Derin Öğrenme, Siyam Ağı, mikro-Doppler, Veri Arttırılması.

Acknowledgement

I gratefully thank my advisor Ömer Morgül for all his guidance and support during my graduate years.

I would like to thank my thesis committee Tolga Çukur and Mehmet Önder Efe for their valuable comments and times.

I want to thank sincerely my dearest family for their love and support throughout my life.

I would like to thank my wife Beyza for his love and understanding, he made my life so beautiful in Bilkent.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview of Activity Recognition

Human activity recognition is an active and challenging task that aims to recognize a person's movements using images or sensor data. Human Activity Recognition (HAR) systems process spatial or/and temporal data using camera and sensor data to understand human behavior. This system has gained more importance due to the growing automation systems and increasing demands from the industry. Therefore, academic researchers give much attention to the HAR system including area of soccer [6], tennis [7, 8], ballet [9], human interaction [10], simple actions [11, 12], healthcare [13], pedestrian traffic [14], and human gestures [15].

On the other hand, there are many applications of HAR in industry, including medical systems, surveillance-based security, and entertainment. In entertainment, some games are designed according to the actions of players [16] that increases the interaction between humans and computers during the game. In addition to entertainment, healthcare and medical systems are other important areas for activity recognition since the actions of patients give critical knowledge

for the rehabilitation process [17]. Furthermore, abnormal activities can be recognizable from surveillance-based security systems thanks to HAR models [18].

1.2 Overview of Classification Tasks for Radar Data

Radars have advantages and disadvantages over the camera. The radar data output is not as clear and understandable as the camera output, but foggy and rainy weather does not affect the operating performance of the radars. Hence, radar data is essential to continue object classification even in difficult weather conditions.

Therefore, radar data is used in many areas for classification tasks. Some of these classifications are Unmanned Aerial Vehicle (UAV) and drones [19], commercial aircrafts [20], armored vehicles [21] and ships [22].

To be able to use radar data in classification tasks, we need to be able to feed these signals as input to classification methods. To achieve this transformation, we need to apply the pre-process. Radar data representations can be obtained after pre-processing the radar data. Radar data representations include the Occupancy Grid Map (OGM), Range-Azimuth-Doppler (RAD) tensor, and micro-Doppler signature.

OGM is one of the 2D representations of radar data. OGM is created as a two-dimensional grid map by using Bayesian Filter [23] to determine whether objects exist at each location. With this map, we can determine the location of obstacles with a bird's eye view. The authors of [24] classify objects, including cars, cyclists, and trucks, using the Occupancy Grid Map. Residual Network (ResNet) [25] could be used as feature extractor. After feature extraction, Faster-Region Based Convolutional Neural Network(RCNN) method [26] is used to classify determined objects.

A RAD map is a format in which radar data is represented in three dimensions. This notation includes the target's range, velocity, and azimuth angle. When the raw radar data's Fast Fourier Transform (FFT) is taken through time, we observe time-dependent distance variations. We take the FFT across chirps which is the increase or decrease in the frequency of radar data over a period of time after the time-dependent FFT is taken. Then, we can observe the velocity changes in the drawing, so a 2D Range-Doppler graph is plotted. After obtaining the Range-Doppler map, if the FFT of the Doppler peaks in each interval bin is taken, a 3D Interval-Doppler-Azimuth plot can be generated. In [27], vehicles are classified using the RAD map. The first two dimensions include Range and Doppler. Then, azimuth information was added as the third dimension. The generated three-dimensional map is fed as input to Long short-term memory (LSTM) for the classification task. Thus, in [27], vehicles can be classified using the RAD map.

Another radar data display is Micro-Doppler Signatures. This map also shows small movements such as rotation, oscillation, and vibration around the moving object. These small movements appear as an additional frequency in the signal from the target. Micro-Doppler Signatures are obtained by Short-Time Fourier Transform (STFT) which is

$$X(m,\omega) = \sum_{n=-\infty}^{n=+\infty} x_n w_{n-m} e^{-j\omega n}, \tag{1.1}$$

where w_n is window size and x_n is the signal to be transformed. FFT is calculated over a long time frame. Thus, the total energies of the frequencies in this time interval can be found. However, the time intervals of the frequencies cannot be known. To localize the frequencies, the time interval is split into small pieces, and a Fourier Transform is applied to each segment. In this way, which frequency occurs in which time interval is found.

Micro-Doppler Signatures were used to distinguish between UAVs and birds. In [28], 9.5 GHz Frequency-Modulated Continuous Wave (FMCW) radar data is used. This signal was converted to Spectrogram using STFT. After obtaining Spectrogram, birds and Unmanned Aerial Vehicles (UAVs) are classified using the Support Vector Mechanism (SVM).

1.3 Human Activity Recognition with Micro-Doppler Data

HAR is another classification method. This method classifies human movements such as walking, running, and crawling. There are many image-based models for human activity recognition tasks [29, 30]. These models use the visual content data of humans to classify activities. Undoubtedly, most computer vision applications, including virtual reality [31], surveillance-based security [32], human-computer interaction [33], and home monitoring [34], are highly related to activity recognition works. Therefore, increasing markets from these applications establish the demands of HAR systems on computer vision.

In addition to image-based models, radars can be used in the classification task. Two main types of radars are Pulsed Wave (PW) and Frequency-Modulated Continuous Wave (FMCW). In Pulse Wave, distance is measured by calculating the delay between transmitted and received signal. On the other hand, FMCW transmits linearly modulated signals, and the difference between transmitted and received frequency measures distance. Hence, FMCW radar can measure correctly the short distance, which is 50-100 meters, unlike Pulsed Wave. The main reason for using FMCW radar instead of Pulsed Wave radar for activity recognition is the ability to measure short-range targets.

FMCW radars have an advantage over image-based systems in poor weather conditions because it gives better results in cloudy weather. Furthermore, radars are more reliable than cameras regarding privacy issues since radars cannot gather images of the private environment, unlike cameras [35, 36, 37, 38].

In addition to privacy advantages, micro-doppler radars are highly compact, low power, and small size [39]. Thanks to the benefits of micro-doppler usage, FMCW classification is used in various applications, including sensing smart environments [40], border control [41], and autonomous driving [42]. Furthermore, according to the walking characteristics of humans, they can be recognized [43, 35, 42], and also armed and unarmed people can be classified with FMCW

radars [44].

The main contributions of this thesis are as follows :

1- Designing four different neural network architectures for Human Activity Recognition (HAR) task and reporting a comprehensive study on comparing various architectures based on Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Siamese Networks.

2- Reporting an extensive analysis on comparing different data augmentation methods, including Mix-Up [45] and Window Cropping [46], and their performances in our network.

3- To the best of our knowledge, this is the first study in which three different radars are used simultaneously as parallel paths for HAR, and these paths are combined with the Siamese Network. The mean Human Activity Recognition accuracy of our proposed Siamese-based architecture reaches **94%**, which is about **2%** higher than results reported in [2].

1.4 Thesis Outline

This thesis is organized as follows. Chapter 2 introduces the conventional methods in the field of time series classification, FMCW radar activity recognition, and data augmentation of radar data. Chapter 3 investigates pre-processing radar data, data augmentation methods, and our proposed neural network architectures. Experiments and results of data augmentation methods and proposed deep learning models are presented in Chapter 4. Furthermore, an ablation study is performed for Attention and Global Average Pooling blocks. In the conclusion chapter, we summarize our proposed model and reconsider the results of the experiments. According to the results of the experiments, we suggest potential improvements for future work.

Chapter 2

Related Work

2.1 Time Series Classification

There are many of time-series data like financial outputs [47], audio [48], patient health metrics [49], and sensors [50]. Therefore, people try to use time-series data for classification tasks [51, 52]. In the 2000s, distance-based models are used to classify time-series data. For instance, the methods based on Dynamic Time Warping (DTW) and Euclidean distance use distance-based methods in solving certain classification problems [53]. For decades, Dynamic Time Warping with the k-nearest neighbors classifier has been used often that the distance can be measured for time series by DTW [54, 55] since DTW allows similar shapes to match in case of out of phase in the time axis.

2.1.1 Deep Learning

On the other hand, deep neural networks have been studied in different domains and developed the performance of models on the solutions in a variety of areas, including Computer Vision (CV) [56], Natural Language Processing (NLP) [57], and speech recognition [58]. There are many reasons for the increase in studies

on Deep Learning (DL). The invention of backpropagation, the development of optimization methods, including Stochastic Gradient Descent (SGD) [59], Adaptive Moment Estimation (ADAM) [60] optimizer, and activation functions like Rectified Linear Unit (ReLU) [61] (Equation 2.1) made DL trainable from computational point of view. Furthermore, the discovery of the effect of Multi-Layer Perceptron (MLP) demonstrated the potential for models with multiple layers to yield good performance. On the hardware side, the reduced train and inference times with the Graphic Processing Unit (GPU) increased the applicability of deep models [62]. ReLU activation function, which is frequently used in DL applications, is given as follows:

$$f(x) = x^+ = \max(0, x). \tag{2.1}$$

Deep learning can be divided into supervised, unsupervised, and reinforcement learning. Observations collected about the concept learned in supervised learning are given to the model as a training set. The desired output values for each sample in the training set are also given as labels. Thus, a relationship is established between the input and the output. The model trained using the established relationship tries to predict the outcome from the input data in future observations. Unsupervised learning, unlike supervised learning, is learning the relationships and structures existing in the data without labeling the data as input-output. It makes inferences about the data using the data samples' distances, neighborhood relations, and densities. Reinforcement learning deals with what actions must be taken to achieve the highest amount of rewards in an environment.

Furthermore, Deep Learning can also be classified based on tasks including Computer Vision and Natural Language Processing. Convolutional Neural Networks (CNNs) [63, 64] which is

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m], \tag{2.2}$$

where $f[m]$ is the input, and $g[n-m]$ is the kernel convolve on input. CNN are often used in the Computer Vision area since these models extract shift-invariant and local features with weight sharing from input frames. This algorithm, which captures and classifies features in images with different processes, consists of different layers. CNN consists of two main parts, convolutional layer, and pooling.

The convolution layer is the first layer that processes the image in CNN algorithms. As it is known, images are matrices consisting of pixels with certain values. In the convolution layer, a filter smaller than the original image size convolves over the image and tries to capture certain features from those images. An exemplary convolutional layer is shown in Figure 2.1. The parameters learned in CNN algorithms are the values in these filters. The model constantly updates these values and begins to perceive the features even better.

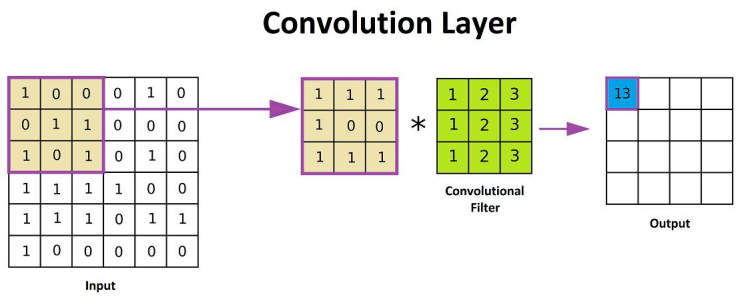


Figure 2.1: Example of Convolution Layer.

Like the convolution layer, the pooling layer aims to reduce dimensionality. There are two different pooling techniques commonly used in CNN models. Maximum pooling takes the largest value in the area covered by the filter, which can be seen in Figure 2.2, while average pooling takes the average of the values in the filter. These layers reduce the size and processing power and preserve essential features.

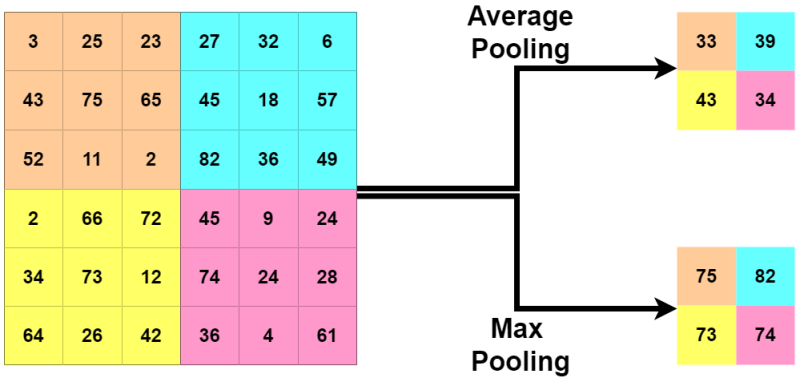


Figure 2.2: Example of Max-Pooling and Average-Pooling.

On the other hand, in Natural Language Processing, recurrent neural networks

(RNN) models [65] are popular in extracting the temporal relations between time-dependent steps. Equations of RNN are given as follows:

$$h_t = \sigma_h(W_h x_t + U_h y_{t-1} + b_h), \tag{2.3}$$

$$y_t = \sigma_y(W_y h_t + b_y), \tag{2.4}$$

where x_t is input vector, h_t is hidden layer vector, W_h, U_h, and b_t are parameter vectors, and σ_h and σ_y are activation functions. RNN always uses the hidden output of the previous step in its step. Hence, it can establish a connection between time steps. For clarity, the unfolded version of the RNN is shown in Figure 2.3 and also equation of its shown in Equation 2.3 and 2.4.

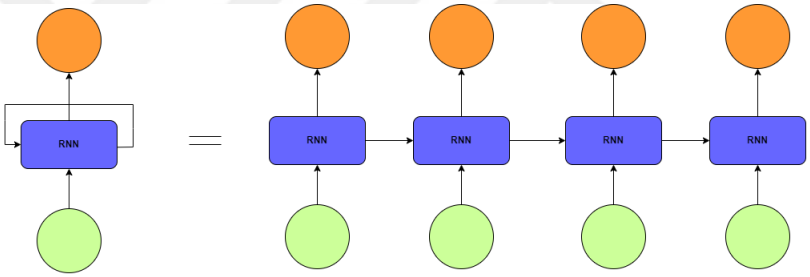


Figure 2.3: Unfolded format of RNN.

Two main types of RNN are the Long Short Term Memory (LSTM) [66] and the Gated Recurrent Unit (GRU) [67]. LSTM has three gates called input, forget, and output gates, which are shown in below equations:

$$f_t = \sigma_g(W_f x_t + U_f c_{t-1} + b_f), \tag{2.5}$$

$$i_t = \sigma_g(W_i x_t + U_i c_{t-1} + b_i), \tag{2.6}$$

$$o_t = \sigma_g(W_o x_t + U_o c_{t-1} + b_o), \tag{2.7}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \sigma_c(W_c x_t + b_c), \tag{2.8}$$

$$h_t = o_t \odot \sigma_h(c_t), \tag{2.9}$$

where x_t is input vector, c_t is cell state vector, W and U are weight matrices, and b_t is bias vector. Also, $\odot$ denotes the element-wise production. Output of σ_g, which is a activation function, creates f_t, i_t, and o_t that are output of forget gate, input gate, and output gate, respectively. To create cell state vector, firstly, input vector is multiplied by weight metric (W_c) and summed with bias (b_c), then tanh

activation function (σ_c) is applied it. In Equation 2.8, element-wise production ($\odot$) of input gate and output of σ_c is summed with element-wise production of forget gate and. Hence, cell state vector (c_{t-1}) is created. Finally, in Equation 2.9, hidden state vector (h_t) is obtained with element-wise production of output gate and activation function output of cell state vector (c_t).

On the other hand, GRU has only two gates called reset and update gates, whose equations are shown below:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z), \tag{2.10}$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r), \tag{2.11}$$

$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h), \tag{2.12}$$

$$h_t = z_t \odot \hat{h}_t + (1 - z_t) \odot h_{t-1}, \tag{2.13}$$

where z_t is update gate vector and r_t is reset gate vector. The update gate behaves similarly to the forget and input gate of the LSTM block. It decides which of the new information will be discarded or added. The reset gate is similar to the forget gate in LSTM. With this gate, it is decided how much information from the past will be kept and discarded.

Block diagrams of Simple RNN, LSTM, and GRU are shown in Figure 2.4. So, LSTM is more complex than GRU. That's why LSTM is preferred for large datasets; GRU is frequently utilized in smaller datasets. These models of RNN can extract short-term and long-term relations between words. Therefore, the performance in the area of speech recognition [68], machine translation [69], and other NLP tasks [70] are improved by LSTM and GRU models.

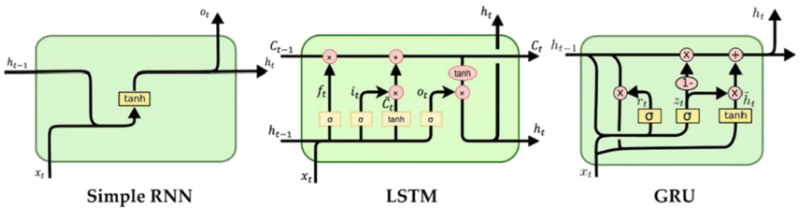


Figure 2.4: Block diagrams of RNN, LSTM, and GRU. This figure is cited by [3].

2.1.2 Time Series Classification with Deep Learning

However growing success of deep learning in different fields, including computer vision and natural language processing, changed the researchers' attention to the deep learning area for time-series classification. At this point, authors of [71] studied deep convolutional neural networks on multivariate time series for classification tasks. After [71], Fully Convolutional Networks (FCN) [72] and Residual Networks(ResNet) [25] are proposed in [73] that brings the state-of-the-art results for time series classification. Then, the concatenation of the LSTM and FCN is suggested in [74]. Therefore, features of these blocks are merged to enhance the classification performance on time series. Long- and Short-term Time-series Network (LSTNet) [75] brings significant progress for time-series classification that uses CNN and RNN to extract short and long time patterns from time-series data. Therefore, they solve the scale-intensive issues of neural network architectures.

In addition to all explained deep learning models, transformer-based models are used to classify time series data. Self-attention mechanisms are used to learn complex patterns of time series data [76]. Furthermore, the authors of [77] propose a novel attention mechanism [78] that uses a self-attention on features instead of time steps. Therefore, more importance is given to features that had a more significant impact on the correct result. Gated Transformer Networks (GTN) [79] is another transformer-based model for multivariate time series classification problems. In [79], Transformer Network is gated to merge the transformer models of channel-wise and step-wise correlations.

2.2 Classification with FMCW Radar

2.2.1 CNN-based Models

Deep learning algorithms in CNN-based models have demonstrated significant development in various studies, including object detection [80], image classification

[81], and image reconstruction [82]. Impressive deep learning results in CNN-based models change the FMCW radar research in the deep learning domain. Therefore, object detection [83, 84, 4], object classification [85, 5], and human activity classification [86, 2] have been studied in the field of FMCW radar data. This section investigates problems and results of FMCW radar studies in the CNN-based models.

One of the studies using FMCW radar is done for the object detection task [83]. This study merges the Occupancy Grid Maps (OGM) and doppler information to classify the moving detections with radar sensors. In OGM, real and false moving detections are distinguished.

Another CNN-based object detection model is given in [84]. It uses Range-Azimuth-Doppler maps to plot 2D boxes on the detected object. The backbone of this architecture is ResNet [25], and the head is You Only Look Once (YOLO) [87].

Range-Doppler-Azimuth radar data is also used by authors of [4] for object detection. Furthermore, a radar target is also used in this research. The radar target displays the radar echo, so the position of the reflecting surface can be shown. By using position information of the radar target mechanism, the 3D block is cropped around each target. This cropped 3D block and target-level features are fed to the proposed RTCNet architecture. This proposed method gives object clustering and classified radar target that can be seen in Figure 2.5.

In addition to object detection tasks, object classification is another research area for FMCW radar inputs. The proposed model of [85] determines the position of the target on range-velocity using Ordered Statistics Constant False Alarm Detector (OS-CFAR) [88] and then implements CNN and fully connected layers to find the object classes, including car, bicycle, and motorbike.

Another object classification model is proposed in [5]. In this paper, there is no localization of the target location as the system considered in [85], but it uses two different types of input to classify motorcycle, car, and person objects. One

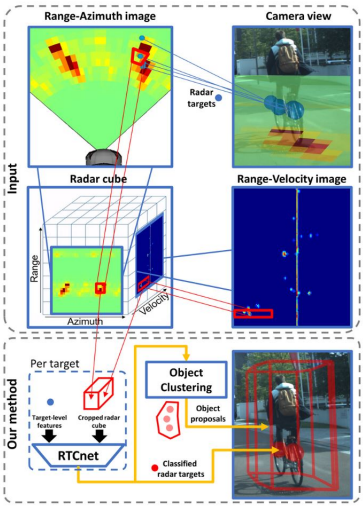


Figure 2.5: Pipeline of RTCNet Architecture. Adapted from [4].

of the inputs is the range-doppler map, and the other is the point cloud map. The Range-Doppler map displays the range-speed response pattern of the target that FMCW radar data can obtain this pattern. On the other hand, the point cloud map shows lidar data as a virtual map. There are parallel CNN-based paths for these two inputs. After extracting features of the Range-Doppler map and point cloud map by CNN-based layers, extracted features are concatenated to merge layers successively. After the concatenation process, CNN and fully connected layers are used for the classification task. Block diagram of this neural network model is shown in Figure 2.6. The range-doppler map and point cloud map do not have common features. Therefore, using different features of the target increases the object classification performance.

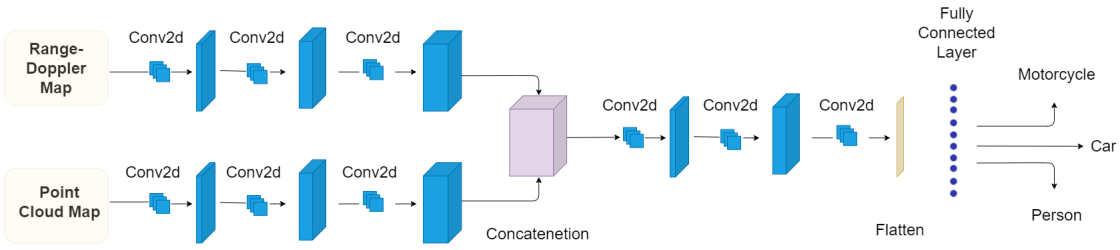


Figure 2.6: Pipeline of model proposed in [5].

Human Activity Recognition (HAR) is another research area for FMCW radar inputs. In [2], the dataset is collected for HAR from three different frequencies,

including 10GHz, 24GHz, and 77GHz, using FMCW radar. This collected dataset is used for cross-frequency training. This article tries to achieve high results by using datasets collected with FMCW radars operating at different frequencies during training and testing. For example, it trains its model with a dataset collected with a 77GHz radar and tests it with a dataset collected with a 24GHz radar. Generative Adversarial Network (GAN) [89] is used to create synthetic data for pre-training their models that classification is made with Convolutional Autoencoder (CAE) model.

Another work that utilizes CNN architecture for the field of Human Activity Recognition with FMCW radar is given in [86], where the raw data is pre-processed to convert it to Spectrogram and Range-Doppler map. By using Principal Component Analysis (PCA), the dimensions of plots have been reduced. After dimension reduction, the classification between people walking and running could be made with the Support Vector Machine (SVM) method.

2.2.2 Recurrent-based Models

FMCW data can also be used with recurrent-based models since there is a relation between temporal steps of radar data and recurrent-based models, including LSTM, GRU, Convolutional LSTM (C-RNN) [90], and Convolutional GRU (C-GRU) [91] extract time-related features of input signal. Therefore, using recurrent-based models instead of CNN-based models improves classification performance for time-dependent inputs. Since FMCW radar data is also a time-dependent input, recurrent-based models have been used in many studies [27, 92, 93] and [94].

In [27], the authors combine CNN-based and LSTM-based methods. Firstly 3D Range-Azimuth-Doppler (RAD) plot is obtained using similar pre-processing methods with the model of [86]. Then, the time series features are extracted with the RAD graph. The LSTM blocks use the temporal information of the features extracted from the RAD plot. Finally, object detection is completed with SSD model [95].

For Human Activity Recognition with FMCW radar, authors of [92] propose stacked Bidirectional LSTM (Bi-LSTM) network instead of LSTM blocks. Therefore, they can utilize bi-directional temporal information from human activation. The proposed model can capture forward and backward temporal information thanks to Bi-LSTM. Therefore, the forward and backward characteristics of the target's movement can be used for classification.

Recurrent-based models are also used for human activity classification tasks. In [94], authors try to classify home activities, including washing, walking, vacuuming, and sedentary. They use FMCW radar data as input converted to a Joint-Time-Frequency (JTF) signature to achieve the classification task. The classification process is carried out by applying the GRU model to the 2D converted input data.

In addition to LSTM and GRU, Convolutional LSTM (C-LSTM) [90] and Convolutional GRU (C-GRU) [91] are another recurrent model to extract temporal features that these recurrent models are fed to 2D input like Range-Doppler spectrogram. Authors of [93] use C-LSTM and C-GRU to extract features of moving targets. The main advantage of using C-LSTM and C-GRU is significantly decreasing the number of parameters that is proper for memory-constrained applications. Otherwise, the number of operations is increased. In [93], there are three classes: pedestrian, bicycle, and vehicle. For CNN-based models, it is difficult to distinguish between cyclists and pedestrians, but the C-LSTM and C-GRU models can classify them correctly.

2.2.3 Dataset

The radar dataset is difficult to standardize because FMCW radars have many configurable parameters. In studies on FMCW radar data, it is difficult to compare the results because the parameters of the dataset can differ.

Furthermore, labeling radar data for object detection is difficult for humans due to the low readability of FMCW data. So, manual annotations can cause

reasons for wrong labeling. To solve this problem, auto-annotation is applied on FMCW data by using stereo cameras in [4], [84] and [96].

In [4], the authors utilize a pair of stereo cameras for labeling. In addition to the study in [4], labels are annotated in the Cartesian form, and also all dimensions of Range-Azimuth-Doppler data are placed in [84]. They utilize radar and stereo instances for auto-annotation. On the other hand, authors of [96] use an unsupervised pretraining algorithm for detecting moving objects in FMCW data with little ground-truth labels. Synthetic Aperture Radar (SAR) images and a pre-trained CNN-based model are utilized to generate labels automatically.

In addition to object detection tasks, there are also HAR radar datasets which are [97] and [2]. Since we try to classify human movement using FMCW radar data in our thesis, we test two datasets on the proposed models. The configuration parameters of the radar used in [86] and [2] are different. Hence, the models of these articles are not comparable. At this point, we trained and tested our proposed models with both of them. Thus, we analyzed the results separately.

2.2.3.1 Dataset 1

Authors of [97] collect human activity dataset to distinguish fast and slow movements from FMCW radar that the center frequency of the used radar is 77 GHz. In order to avoid confusion between datasets, the name of this dataset was defined as Dataset 1 [97]. Experiments are performed on people who walk at different speeds in front of a single radar without any obstacle. There are three classes in Dataset 1 [97]. These are "slow", "slow on pocket" and "fast". There are 57 samples in total for each class. These samples are gathered from 19 different humans. The data structure is given in Table 2.1.

When creating Dataset 1, it was designed by the authors of [97] for a total of 3 classes, but we combined the classes "slow" and "slow with pocket". This is because both classes' Spectrogram and Range-Doppler plots are very similar. In terms of the data produced by the radar, there is no difference between whether

Activity	No of Subjects	Number of Repetition	Number of Tests
Fast	19	3	57
Slow	19	3	57
Slow Pocket	19	3	57

Table 2.1: Details of Dataset 1 [97].

the target person has his hand in his pocket or not. Therefore, our experiments have 57 "fast" samples and 114 "slow" samples that samples of these classes can be seen in Figure 2.7.

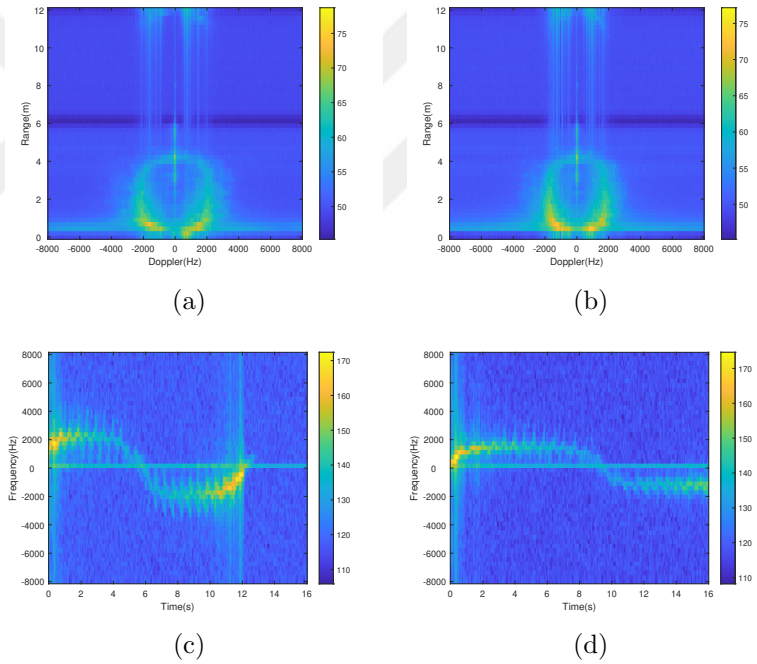


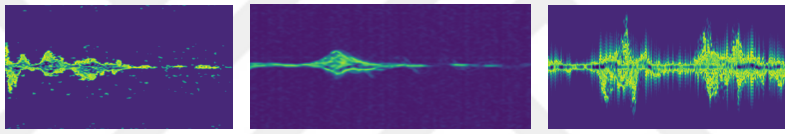
Figure 2.7: Range-Doppler samples of a person walking (a) fast and (b) slow. Spectrogram samples of a person walking (c) fast and (d) slow.

2.2.3.2 Dataset 2

In Dataset 2, three different radar is used to collect FMCW radar data in [2]. There are three different FMCW radars in this dataset. The first one is the Texas Instruments IWR1443 FMCW transceiver. The center frequency of this radar is 77 GHz, and its bandwidth is 750 MHz. The second one is the Ancortek SDR-Kit.

Its center frequency is 24 GHz, and it has a 1500 MHz bandwidth. The final one is the XeThru X4 sensor transmitting across a band of around 7 GHz - 10 GHz. Samples of these radars can be seen in Figure 2.8. Furthermore, all used radars are placed side by side. In this way, all radars can observe the same scene.

Dataset 2 [2] contains a total of 11 classes. These are "walking towards radar", "picking up an object from the ground", "sitting on a chair", "crawling towards the radar", "walking on both toes", "scissor gait", "walking away from the radar", "bending", "kneeling", "limping with right leg stiff", and "walking with short steps".



(a) (b) (c)

Figure 2.8: Spectrogram examples of a person bending in front of the radar at (a) 10 GHz, (b) 24 GHz and (c) 77 GHz radar data.

The FMCW radar outputs of some classes in Dataset 2 are very similar. For example, "bending", "kneeling" and "picking up an object from the ground" are very similar actions when viewed from the radar. The difficulty of the task increases as there are very similar classes.

Before converting raw data from radars to Spectrogram, pre-processing is done. The pre-process of Dataset 2 is shown as a block diagram in Figure 2.9. Furthermore, all the steps of the applied pre-process were explained in order:

1. The raw data is validated by applying phase corrections on in-phase and quadrature (I&Q) channels to ensure that the two baseband signals are orthogonal to each other with the same amplitude.

2. 1D signal is converted to 2D.

3. The Moving Target Indication (MTI) filter is then applied to the 2D output to find moving objects.

4. The Short Time Fourier Transform (STFT) is applied to obtain the Spectrogram.

Figure 2.9: Block diagram of pre-process of Dataset 2 [2].

2.3 Data Augmentation

Deep learning models give outstanding results for time-series tasks. But, modern deep learning models are data-hungry for the purpose of training. Fewer data can cause over-fitting problems. However, there is limited labeled time series data for deep learning architectures. At this point, data augmentation has a significant role in increasing the size of time-series data.

On the other hand, the data augmentation task is more complicated for multivariate time series data since there can be complex relations between features of time-series data across time. So, data augmentation methods for image processing may not give adequate improvement for time-series tasks. Furthermore, the data augmentation techniques may not be used for all time-series tasks like classification or object detection.

There are two types of data augmentation methods for time series data. The first one is the basic approaches that include window cropping [46], MixUp [45], permutation, rotation, scaling [98], and masking blocks of spectrogram [99]. The second one is the advanced approaches, including learned latent space [100], embedding space augmentations [101], Generative Adversarial Networks (GAN) [102, 103], Recurrent Conditional Generative Adversarial Networks (RCGAN), and Deep Reinforcement Learning (DRL) [104].

One of the basic approaches is window cropping, which is to apply window cropping to time series data over time [46]. During training, cropped windows

are selected randomly. On the other hand, each window from test time-series data is used with the majority voting method for the classification task during the test model. The MixUp Augmentation [45] method is often used for image-based models [105, 106]. However, since this method is based on the convex combination, it can also be used for time series data. Here, to increase the number of data, the weighted average of two samples is taken, and a new sample is created. The weighted average of the labels is also taken at the same rate. Hence, new samples added to the dataset increase the diversity of the data set. [98] combines permutation, rotation, and scaling processes to enhance the size of the dataset. Unlike previous works, [99] utilizes spectrograms of radar data that apply masking blocks of time steps and frequency channels.

The authors of [100] propose that data augmentation be implemented in the learned latent space. In [101], latent space augmentation is also used, but model of [101] trains a classification model with various compositions of embedding space augmentations. In addition to latent space augmentation, generative time-series models are used. For example, Recurrent Conditional GAN (RCGAN) is used for time-series data augmentation in [103]. Furthermore, there are studies on automated data augmentation for time series [104]. They utilize a reinforcement learning framework to search for effective data augmentation methods on automatic.

Chapter 3

Model Architectures

3.1 System Description

As explained in ([97]), Texas Instruments mmWave radar is used to obtain the Dataset 1 [1]. This radar, owning one receiver and one transmitter, is developed for the automotive car market. Frequency Modulated Continuous Wave (FMCW) modulation is used to transmit a signal whose frequency ranges from lowest to highest value in a time gap called chirp time. The FMCW Radar works in 76-77GHz and 77-81GHz frequencies with one and 4 GHz bandwidths. The first one is used for long-range applications (up to 150 meters), and the second one is used for short-range applications (up to 30 meters). The radar operated in this system is a Software Defined Radio (SDR). It works in the frequency range 24-26 GHz, and a chirp time and maximum ramp slope of 15.625 MHz/μs, with a maximum bandwidth of 2 GHz.

3.2 Range-Doppler Input Pre-Processing

In Dataset 1 [1], there are four channels. We have summed these channels and acquired a receiver channel to enhance the signal-to-noise ratio. After summing channels, we have reorganized this vector into a matrix. The rows of this matrix are calculated from a single chirp, called fast time. On the other hand, the matrix columns are computed from various chirps, named slow time. By using this matrix, the Range-Doppler map can be obtained. First, we apply Fourier transform to matrix columns to get the Range-Doppler map. After applying Fourier transform to columns, the Fourier transform is applied to the rows of the matrix. Furthermore, the block diagram is shown in Figure 3.1 to make the Range-Doppler pre-processing more understandable.

As clarified in ([107]), the beat signal, which is the difference between the transmitted signal's carrier frequency and the received signal's carrier frequency, consists of M targets, and a single receiver within period T can be described as:

$$s(l,n) = \sum_{m=1}^{M} a_m exp(j(2\pi(f_{r,m} n T_s + f_{d,m} l T + \gamma_m))) + w_s(l,n) \tag{3.1}$$

In Eq. (3.1), f_s is the sampling frequency, T_s is the sampling period, and $N = T/T_s$ is the number of samples, $n = 0, 1, \ldots N-1$. Furthermore for $m = 1, .., M$, $f_{d,m}$ is the mth (target) Doppler frequency,$f_{r,m}$ is the range frequency, γ_m is the Range-Doppler coupling, and finally $w_s(l,n)$ is the noise. In this beat signal, there are two indices, which denote the fast time index n and the slow time index l. In order to estimate the range information, L times N_S-point range-FFTs across the fast time are performed on the beat signal sequences $s(l,n)$, where N_S is the number of range-FFT points. Therefore, the kth range-FFT output of the lth sampling sequence $s(l,n)$ can be expressed as:

$$S(l,k) = \sum_{n=1}^{N_s} s(l,n) exp[-j\frac{2\pi}{N_s} n(k-1)] \tag{3.2}$$

In Eq. (3.2), k is the index of range-bins in the range domain, for $1 \leq k \leq N_S$. At the same time, in order to estimate the velocity information, N_S times N_C point Doppler-FFTs across the slow time are performed on $S(l,k)$. The qth Doppler

FFT output is expressed as $\bar{S}(q,k)$, as follows:

$$\bar{S}(q,k) = \sum_{l=1}^{N_C} S(l,k) exp[-j\frac{2\pi}{N_C}(l-1)(q-1)] \tag{3.3}$$

where q is the index of Doppler-bins in the Doppler domain and $1 \leq q \leq N_C$, where N_C is the number of Doppler-FFT points.

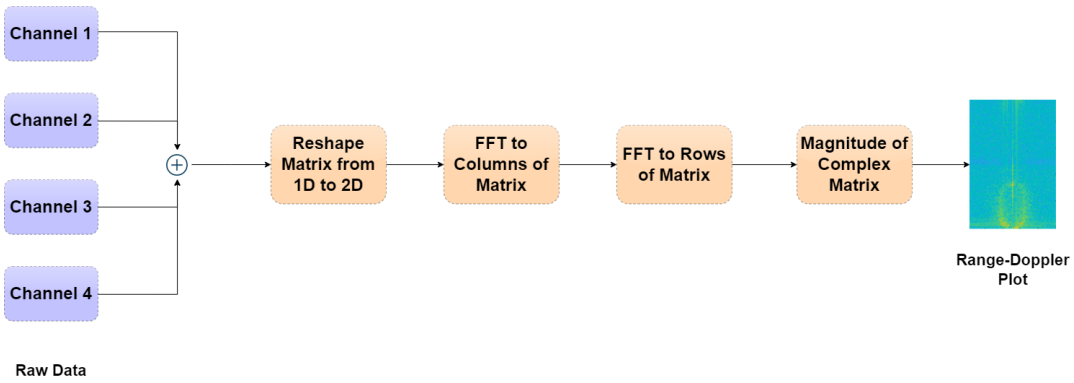


Figure 3.1: Block diagram of pre-process of Dataset 1 [1].

3.3 Spectrogram Input Pre-Processing

In addition to the Range-Doppler map, Spectrograms can be obtained by processing raw radar signals. Figure 3.2 shows the sample input and sample output of the Spectrogram pre-processing. The spectrogram is a graphic of the time-varying frequencies of a signal. It can be representable as a frequency-time map. The spectrogram is derived by taking the Fourier transform of the split time-domain signal $x[m]$. This technique can be called a Short Time Fourier Transform operation and is described as:

$$X_n(e^{j\omega}) = \sum_{m=-\infty}^{\infty} w[n-m]x[m]e^{-j\omega m} \tag{3.4}$$

In Eq. (3.4), $w[n-m]$ is a window sequence. Generally, the Gaussian or Hann windows centered around zero are used. $x[n]$ is the time-domain radar signal sampled at the sampling frequency. We must distinguish the frequency shifts, so the window size should be wide enough. The Spectrogram is the magnitude

square of the power spectral density that is formalized as $|X(e^{jw})|^2$ found in Eq. (3.4). We used Hann Window and a window size of 512 samples for the STFT Equation in our configuration.

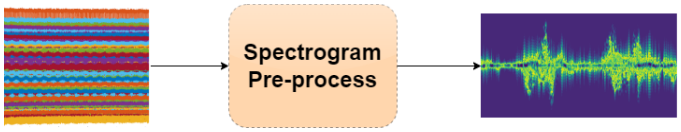


Figure 3.2: Input and output of Spectrogram pre-process.

3.4 Normalization

Normalization is a pre-process for training when the features in the data have different ranges to set the values of those features to a similar scale. Normalization has many advantages. Some of these are as follows:

- Reduces internal variable drift to improve training performance.
- Scales each feature to a similar range to reduce neural network bias.
- Stabilizing the gradient descent enables a greater learning rate to be used.
- Speeding up the optimization process by limiting the weights to a certain range.

We pre-processed all the spectrograms and range-doppler maps obtained to apply the normalization process. In this process, the mean value of all images converged to 0, and the standard deviation was determined as 1. Equation 3.5, which is

$$I_{norm} = \frac{I - I_{mean}}{I_{variance}}, \tag{3.5}$$

was applied to all maps to use the normalization process.

3.5 Data Augmentation Methods

There is a shortage of FMCW radar data for academic research. Also, since each FMCW radar data has its configuration features, different datasets cannot be combined and used. On the other hand, we need a lot of data to train deep learning models. In this context, we need to increase the number of datasets to properly train our deep learning models. One of the best ways to do this is to use Data Augmentation methods. With these methods, data redundancy can be avoided, data diversity can be increased, and model prediction accuracy can be increased [108].

In the thesis, we will use two data augmentation methods to increase the number of spectrogram and range-doppler data generated from FMCW radar data. The first is the MixUp Augmentation method, which is also used to increase the number of image data [45]. The second method is the Window-Cropping method [46].

3.5.1 Mix-Up Augmentation

Dataset 1 [1] has a total of 171 samples. 4 out of 5 of the whole dataset is split for training. Therefore, there are 138 samples to train the proposed model, which is inadequate to train the neural network appropriately, which can cause overfitting. Therefore, we need to increase the depth of the dataset.

On the other hand, the Dataset 2 [2] also contains 11 classes, each containing 60 samples for 77 GHz and 10 GHz radars. This figure is also insufficient for training the models we propose.

The Range-Doppler and Spectrogram plots of Dataset 1 [1] and Dataset 2 [2] are regarded as image data. Hence, we used the mixup augmentation method ([109]) to these plots to expand the number of Range-Doppler and Spectrogram plots.

The mixup augmentation technique ([109]) creates new images by pixel-wise mixing up the different images. Furthermore, this method mixes the labels of related images. The mixup augmentation technique can be thought of as the convex combination. Let x_i and x_j be two samples whose labels are y_i and y_j, respectively. The equations of mixup augmentation are

$$\hat{x} = \lambda x_i + (1 - \lambda) x_j \tag{3.6}$$

$$\hat{y} = \lambda y_i + (1 - \lambda) y_j. \tag{3.7}$$

where $\lambda \in [0,1]$ has been picked from a Gaussian distribution whose mean and standard deviation parameter values are hyper-tuned and determined as 0.2 and 0.02, respectively. In this way, a new image $\hat{x}$ and its label $\hat{y}$ may be constructed for the selected pairs. Hence, the newly created images and its label are very different from the dataset we have. Thus, the diversity of the dataset has increased.

An example of this augmentation method is shown in Figure 3.3. Figure 3.3 shows the instance of mix-up augmentation method. Fast and slow samples of the Range-Doppler map are used as input for mixup augmentation. λ is selected as 0.5 for this example. The outcome of this technique is the mixed-up of these given inputs. The label of the output image is 0.5 because λ is set as 0.5.

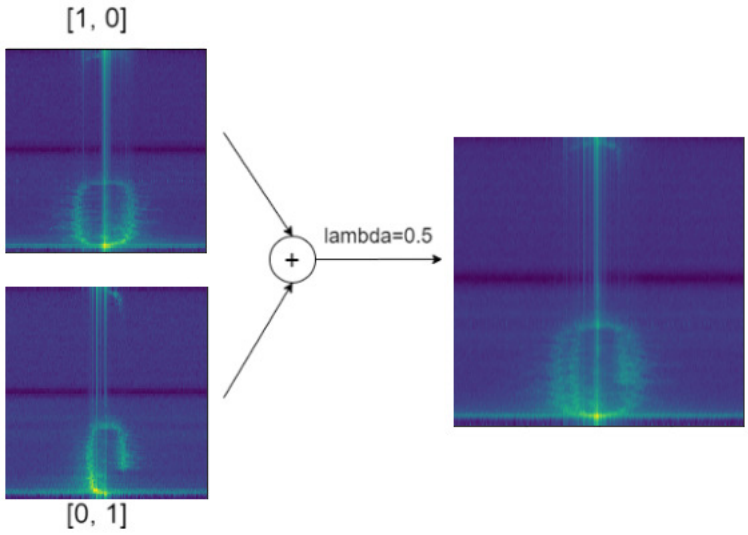


Figure 3.3: Example of Mix-Up Augmentation method.

3.5.2 Window-Cropping Augmentation

Another data augmentation method is Window-Cropping [46]. This method can be used for time series data as the obtained spectrogram is divided into four along the time axis. These split spectrograms are fed into the neural network model separately. In order to understand the method, an example of the Window-Cropping process is shown in Figure 3.4. If we do this for the image inputs, the whole structure of the picture is changed drastically. Therefore, we only performed Window-Cropping experiments on models with time series blocks.

The thing to be careful about while experimenting Window-Cropping Augmentation is that the samples reserved for training are not used on the test side. If separated spectrograms from the same sample are used for training and testing, the results will be better than in real-life scenarios. Therefore, samples are first separated for testing and training, followed by dividing.

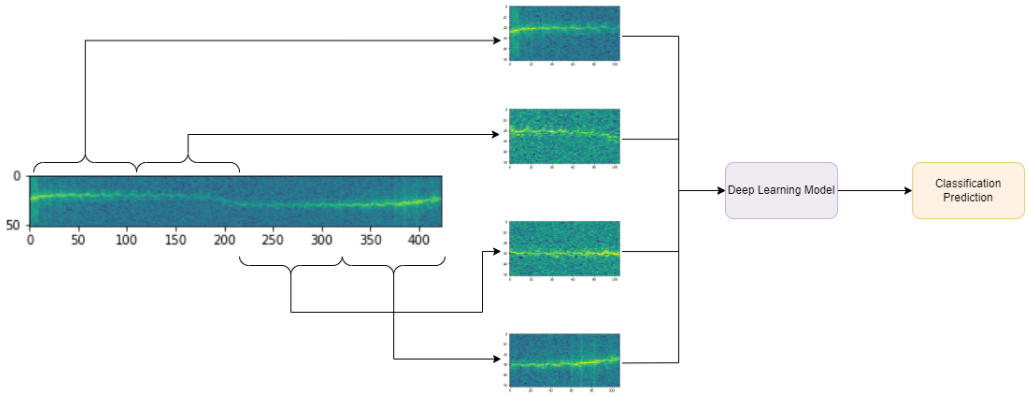


Figure 3.4: Example of Window-Cropping Method.

The Window-Cropping method performs the testing phase with a majority of votes. Each quartered sample is tested separately on the model, and the classification results of the four split spectrograms are summed. The human activity class of each sample is estimated using the maximum value between classes.

3.6 Attention Mechanism

In the typical Attention Mechanism, each time step above the input is multiplied by the W coefficient, summed with the bias, and passed through the activation function. After this process, the activation function's output is passed over softmax. Then we multiply the Softmax output again in steps. Therefore, the steps that contribute better to the result are multiplied with higher weights. After multiplication, outputs continue to be used in the neural network model. The equations of Attention Mechanism are

$$e_{ij} = \sigma_g(W_i x_j + b_i), \tag{3.8}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}, \tag{3.9}$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} x_j \tag{3.10}$$

where x_j is input vector, W_i is weight metric, b_i is bias vector and σ_g is tanh activation function.

On the other hand, instead of establishing a time-dependent attention structure, we set a feature-dependent attention structure for Spectrogram inputs. To do this, transpose of the input was used before feeding the Attention model. Hence, the Spectrogram features, which contribute more to the reduction of loss, are multiplied by more weight.

3.7 CNN-based Activity Recognition

Our first proposed method is CNN-based activity recognition architecture that uses input Spectrogram plots as images. So, this architecture uses CNN-based neural network blocks, including a convolutional neural network. The model in Figure 3.6 uses only the Spectrogram input, while the model in Figure 3.5 uses both the Spectrogram and Range-Doppler in parallel. The reason we tried both

models based on CNN is to see the effect of the extra Range-Doppler input on test accuracy results. This comparison is made in Table 4.18.

Firstly, Spectrogram plots are normalized to prevent dataset variance and scaling effects.

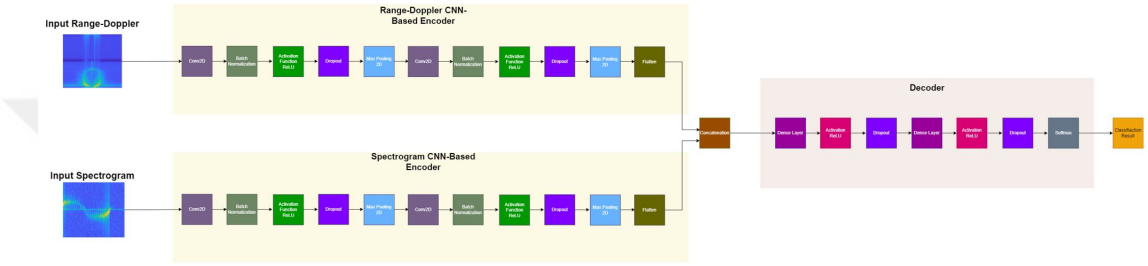


Figure 3.5: Dataflow of our proposed CNN-based architecture for Dataset 1 [1].

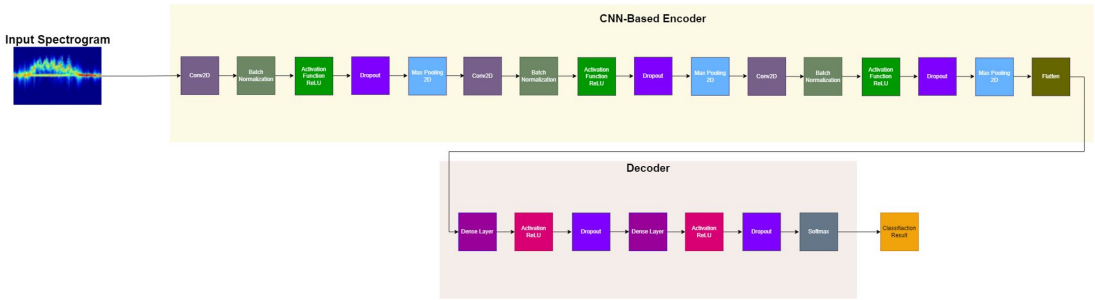


Figure 3.6: Dataflow of our proposed CNN-based architecture for Dataset 2 [2].

After normalizing input plots, these are processed with a convolutional neural network block. Then, Batch Normalization is applied to prevent overfitting. After batch normalization, the Activation function, which is ReLU, is implemented for getting nonlinear results. A dropout layer is added to avoid memorization during training. Finally, the max-pooling layer is implemented. Therefore, the dimension of the image is decreased by half throughout the vertical and horizontal axes.

These explained blocks were repeated three times to extract features deeply. Finally, 2D results are flattened. Softmax can be applied for classifying activity recognition thanks to the flattening process.

3.8 LSTM-based Activity Recognition

As we did with the CNN-based model, we tried different models in the LSTM-based model, where only the Spectrogram is fed into the model, and the Range-Doppler and the Spectrogram are fed together to the model. Only the model with both inputs fed together is shown in Figure 3.7 to avoid duplication.

Our second proposed method is the LSTM-based neural network architecture that uses Range-Doppler and Spectrogram plots as input for activity recognition. These images are normalized before feeding the neural network to eliminate scaling and variance effects. Normalization is applied to input images to eliminate scaling and variance impacts before feeding the model.

The proposed architecture uses the Bi-LSTM layer instead of the CNN layer. The Bi-LSTM layer constructs a directed graph bidirectionally along with a temporal sequence. Therefore, the Bi-LSTM layer can extract the time-dependent features of Range-Doppler and Spectrogram images.

Additionally, instead of the LSTM layer, the Bi-LSTM layer is selected because the Bi-LSTM also uses the reverse time sequence. By using bidirectional LSTM, the reverse flow of time feature is also extracted.

The proposed model consists of three main blocks. The first is the encoder part, the second part is the concatenation, and the third one is the decoder part. Furthermore, the encoder part consists of two different paths that Spectrogram and Range-Doppler plots are used as input for these paths. Parallel paths have the same layers and parameters. Two paths of the encoder part are combined with the concatenation part. The features from both paths for concatenation are listed one after the other. Hence, all outputs of the routes can be used.

In the encoder part, the first block is the Bi-LSTM. There are different frequency bins in the vertical axis of the Spectrogram shown in Figure 2.7. These frequencies are used as input of the feature of the Bi-LSTM input. On the other side, the time steps of the Spectrogram are represented on the vertical axis of

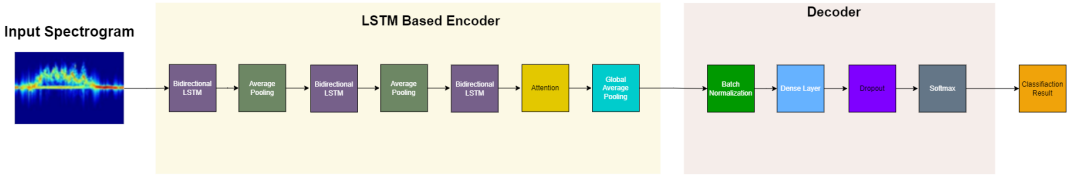


Figure 3.7: Dataflow of our proposed LSTM-based architecture.

the Spectrogram. The Bi-LSTM layer uses the relation between the amplitude of FMCW radar through time sequence for each frequency.

On the other hand, in the parallel encoder part, range values are used as feature input to Bi-LSTM for Range-Doppler plots. The horizontal axis of the Range-Doppler images shows the object's speed, and the vertical axis is the range. So, Bi-LSTM uses the speed sequence and range values as the feature. Hence, thanks to parallel encoder paths, we can use extra features from these plots and get better results than one input case, see Table 4.18.

In order to prevent overfitting, regularization blocks including Batch Normalization ([110]), and Dropout ([111]) layers are implemented after the Bi-LSTM block. Then, outputs of parallel encoder paths are concatenated to feed the decoder part.

Two dense layers have used that activation for the decoder part, and dropout layers follow. The binary cross-entropy loss function is used for binary classification at the end of the decoder part.

LeakyRelu ([112]) is used for all activation functions. Besides, adam optimization ([113]) and early stopping ([114]) methods are used to get better accuracy results.

3.9 LSTM- and GRU-based Activity Recognition

We also propose another method: LSTM- and GRU-based activity recognition. This method uses stacked LSTM and GRU blocks in parallel. Spectrogram plots that we use as inputs are fed to both paths. In this way, we can use different features that LSTM and GRU blocks extract from the Spectrogram plot together.

Before feeding Spectrogram plots to the proposed neural network architecture, the normalization process is applied to eliminate dataset scaling and variance drawbacks.

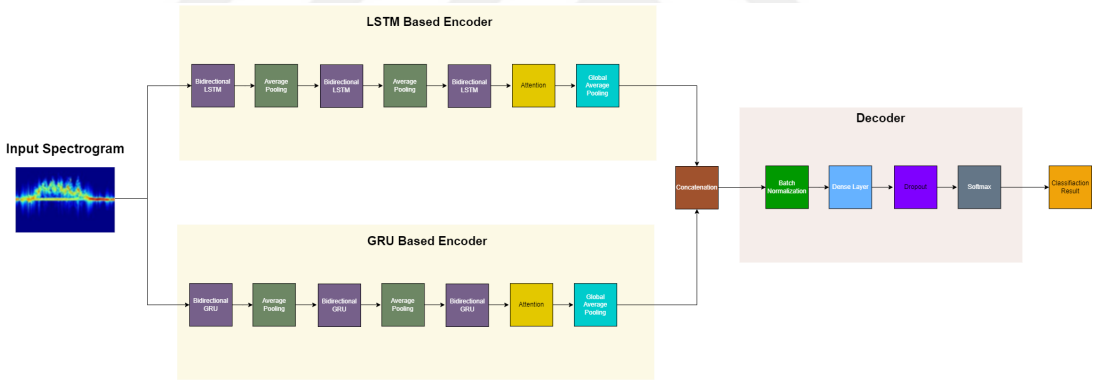


Figure 3.8: Dataflow of LSTM- and GRU- based architecture.

There are three stacked bidirectional GRU and LSTM for each path. Average pooling is settled in between stacked GRU and LSTM blocks. Therefore, parameter size is diminished, and longer time interval is used by time series blocks, including GRU and LSTM.

The main advantage of using GRU and LSTM in parallel neural network paths is that GRU and LSTM search different time intervals to find time dependence between them. Hence, additional features are extracted during these parallel paths.

After using stacked time series blocks, attention is implemented to giving more weight to related features. Each feature has a different impact on the classification

Figure 3.9: Dataflow of our proposed LSTM- and GRU-based architecture for Multi-Frequency input.

task. Providing the same weight to these features diminishes the features' effect. On the other hand, giving greater weight to features that provide better test accuracy results creates better classification results. The ablation study in Table 4.16 was performed to show the effect of attention blocks on the test accuracy outcomes.

At the end of the paths, the global average pooling method takes the average of the features for whole time steps. The effect of global average pooling on test accuracy results is shown in Table 4.17. Concatenation is applied to diminish the number of paths from two to one. Hence, Concatenation can merge the outputs of the two paths.

After concatenating two outputs, batch normalization is implemented to prevent overfitting. Then, dense layers and dropout layers have used. Finally, the categorical cross-entropy loss function is used for softmax classification. Therefore, neural network architecture is completed for the classification task of activity recognition.

On the other hand, Dataset 2 [2] contains FMCW radar data from three different frequencies. Each of these frequencies contains different characteristics of the target. Therefore, the simultaneous use of three radars can improve performance in classifying human movement. At this point, we fed the dataset at three different frequencies in parallel with the model we created. This pattern can be seen in Figure 3.9. In this way, the increase in our performance can be seen in Table 4.22.

3.10 Siamese Multi-Frequency Activity Recognition

The final proposed method uses three different Spectrogram plots having different bandwidths. These three plots are trained throughout six different paths. Three of them are based on LSTM, and the others are based on GRU. LSTM-based paths

share weights within themselves, and GRU-based networks also share weights within themselves. Hence, these paths can be mentioned as Siamese networks. Figure 3.10 was used to visualize the proposed model. All weights are the same among the blocks defined as "Tied Weights" in this figure.

The same normalization methods as previous methods are applied to input plots for scaling and variance issues.

Figure 3.10: Dataflow of our proposed Siamese-based architecture.

GRU and LSTM encoder paths in Siamese-based are similar to LSTM- and GRU-based network architectures. However, the Siamese-based model uses three different bandwidth Spectrogram inputs, 10 GHz, 24 GHz, and 77 GHz. Each of these inputs is fed through separate paths, but all LSTM blocks and GRU blocks use the same weights within themselves.

Three different radars observes to same scene but all of them gives different results from the same scene because of the different bandwidths that can be seen

in Figure 2.8. Hence, different features can be extracted by using different radars.

On the other hand, the Siamese-based network gives better results since networks need fewer parameters for training. So, the possibility of overfitting is decreased. All comparisons including test accuracy results of different models are shown in Table 4.22.

Chapter 4

Experiments

4.1 Performance Metrics

Dataset 1 [1] consists of two classes. These are classes of slow and fast walking people. There are 114 examples of slow-walking people. On the other hand, the number of fast walking samples is 57. Dataset 1 [1] has an unbalanced class distribution due to the sample size difference between classes. Therefore, in addition to the accuracy metric, F1-Score is used as a performance metric ([115]). F1-Score consists of Recall and Precision metrics. Precision is the ratio of true positive samples to all predicted positive samples [116]. On the other hand, Recall is the ratio of true positive samples to all expected positive samples. To describe these, we first define TP (True Positive), TN (True Negative), FP (False Positive), and FN (False Negative) values. Precision and Recall can be expressed as follows:

$$Precision = (TP)/(TP + FP) \tag{4.1}$$

$$Recall = (TP)/(TP + FN) \tag{4.2}$$

F1-Score is the harmonic mean of Precision and Recall. So, if Precision or Recall has a small value, F1-Score will also have a small value.

$$F1\ Score = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} \tag{4.3}$$

4.2 Training Setups

There are four different neural network architectures in our research. These are CNN-based model, LSTM-based model, LSTM- and GRU-based model, and Siamese-based model that architectures of these models can be seen in Figure 3.6, 3.7, 3.8, 3.10, respectively. Parameters of these models are tuned to get the most desired results in the case of classification, training time, inference time, and minimum standard deviation from three repeated run experiments. Mentioned experiments are performed with the Google Colab Pro application. This application supplies NVIDIA Tensor Core GPU or Tesla P100 GPU. Tuned parameters of all of the proposed models are shown in Table 4.1, 4.2, 4.3 and 4.3.

All values of dense sizes and dropout rates were obtained from systematic experiments by grid search. We set the batch size as 32. We use a small batch size to not fill up the RAM capacity during training. Also, we chose the batch size of 32 because it is a power of 2. Therefore, we can get maximum efficiency from GPU [117]. Since the batch size is selected small, the learning rate is also chosen small. Furthermore, the activation function is selected as Leaky Relu instead of Relu to avoid the "dying Relu" problem [118]. Equation of Leaky Relu is

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ 0.01x & \text{otherwise,} \end{cases} \tag{4.4}$$

that if input signal x is smaller than zero, it is multiplied by 0.01.

Also, we used the Categorical Cross-Entropy loss function, whose equation is

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}), \tag{4.5}$$

because our research in this thesis is to classify human motion, which includes 11 different classes for Dataset 2 [2] . On the other hand, the Sigmoid loss function

is used for Dataset 1 [1] since this dataset consists of two classes, including fast and slow walking.

Parameter	Value
Batch Size	32
CNN Filter Sizes	4-8-16
CNN Dropout Rate	0.1
Dense Size 1	256
Dropout Rate 1	0.25
Dense Size 2	128
Dropout Rate 2	0.3
Dense Size 3	64
Dropout Rate 3	0.1
Activation Function	Leaky ReLU
Loss Function	Categorical Cross-Entropy
Optimizer	Adam
Metric	Accuracy

Table 4.1: Tuned training parameters for CNN-based architecture.

Parameter	Value
Batch Size	32
LSTM Dropout Rate	16
Unit Number of LSTM	128
Dense Size 1	512
Dropout Rate 1	0.25
Dense Size 2	64
Dropout Rate 2	0.15
Activation Function	Leaky ReLU
Loss Function	Categorical Cross-Entropy
Optimizer	Adam
Metric	Accuracy

Table 4.2: Tuned training parameters for LSTM-based architecture.

4.3 Data Split

Before training the neural network models, the dataset is divided into three parts. These are the training, validation, and testing sections. We use the training part to train our model. The validation part is used to measure performance

Parameter	Value
Batch Size	32
LSTM Dropout Rate	16
Unit Number of LSTM	128
Dense Size 1	512
Dropout Rate 1	0.25
Dense Size 2	64
Dropout Rate 2	0.15
Activation Function	Leaky ReLU
Loss Function	Categorical Cross-Entropy
Optimizer	Adam
Metric	Accuracy

Table 4.3: Tuned training parameters for LSTM- and GRU-based architecture.

Parameter	Value
Batch Size	32
LSTM Dropout Rate	0.3
Unit Number of LSTM	128
Dense Size 1	512
Dropout Rate 1	0.25
Dense Size 2	64
Dropout Rate 2	0.15
Activation Function	Leaky ReLU
Loss Function	Categorical Cross-Entropy
Optimizer	Adam
Metric	Accuracy

Table 4.4: Tuned training parameters for Siamese-based architecture.

while training the model. We also use the early-stopping method, thanks to the validation part. This method stops the training process if the validation loss increases as the epoch pass. The testing part is completely separated from the training and validation part. The test data is used to evaluate the model after the entire training process is finished. Our test data must be completely separate from training to create a model that can be suitably used in real life.

The model used the five-fold cross-validation method to obtain reliable results. In each part, all samples are divided into five equal parts, and one of each part is used as the test dataset. This process is repeated five times, and at each part,

the test dataset is selected from a different piece. At the end of these five-folds, the results from all folds are averaged using the mean. Therefore, each sample in the dataset is used for testing, and bias is reduced. In addition, one-fifth of the training data determined in each step is selected as the validation data set. The whole process described above is repeated three times to increase the reliability of the result.

4.4 Systematic Comparisons

The systematic comparisons section is examined under five main headings. These are Architecture Search, Data Augmentation methods, Ablation Study, Dataset 1 [1], and Dataset 2 [2]. In Architecture Search, the minimum, maximum, and mean test accuracy values of different block numbers are compared for each model. In addition, the number of parameters, training, and inference times are also used in comparing the models. Thus, models with different block numbers can be selected for performance and size-oriented studies.

Secondly, the performances of the Data Augmentation methods used to increase the sample size are compared. In order to make this comparison, each of the suggested models is fed utilizing the MixUp Data Augmentation method and Window-Cropping. For the reliability of the comparison, the same model is used for each model in all comparisons.

Another work we have done is the Ablation Study. In this section, we measure the contribution of the Attention Mechanism and Global Average Pooling used in the proposed models in testing accuracy when classifying human movement. To make this comparison, the Attention and Global Average Pooling blocks are removed from the model, and this model is retested.

Finally, comparisons of all proposed models were made separately for Dataset 1 [1] and Dataset 2 [2]. These tables provide an overview of all models for different datasets. Furthermore, all proposed models were tested using each radar (10

GHz, 24 GHz, and 77 GHz) in Dataset 2 under the One-Input heading.

4.4.1 Architecture Search

In this section, we will share the results of our experiments to determine the number of blocks in neural network architectures in tables. All these experiments are run on the same cloud GPUs, Google Colab. Furthermore, all codes and their results are shared on the GitHub page, which is "https://github.com/mertege/Thesis_Experiments".

Block numbers are compared under different headings. The min, max, and mean values of test accuracy are under these headings. All these outcomes are given to examine whether the results obtained during the test are far from each other. Experiments, where min and max are far apart are less reliable because there is a possibility of unsatisfactory results when the model is tested in real life.

We also included the F1 Scores in the table since the binary classification was made in Dataset 1 [1]. As can be seen in the equation 4.3, both the precision value and the recall value must be high for the F1 Score since harmonic averaging is taken. These values are also more appropriate metrics for imbalance datasets than test accuracy [119].

Furthermore, the total number of parameters is also essential in comparing models. The low number of parameters allows neural network models to be implemented on a larger number of devices. Mobile devices, Raspberry Pi and FGPA, are some devices that require small size models. As the usability of these devices increases, their application areas also increase [120, 121, 122]. For this reason, we also included the number of parameters in our comparison tables.

Finally, training time and inference time are added to the experimental tables to compare the models. The training time is an important parameter used during the model development because long training times increase the duration of the experiments. Also, inference time is the time to complete the classification process

on a sample when the model is applied in real life. We can measure the delay in the model's output thanks to the inference time.

4.4.1.1 CNN-Based Activity Recognition

In the CNN-based architecture, conv2d, Batch Normalization, activation function, dropout, and Maxpooling layers are used as blocks, respectively, which can be seen in Figure 3.5. The different numbers of these blocks are stacked to get better classification results.

Experiments were done with different numbers of blocks, and the architecture that gave the best results was searched. These results were analyzed for each title, including test accuracy, number of parameters, training time, and inference time.

In Table 4.5, comparison of different number of blocks can be observed for Dataset 1 [1]. It is observed that the F1 score and test accuracy increase as the number of blocks increases. In addition, since the size decreases with max-pooling in each block, the flatten operation at the end of the CNN blocks and the inputs of the dense layers become small. In this way, our model contains fewer parameters and less training and inference time when more CNN blocks are used. Thus, the best choice is to choose 5 blocks for CNN-based architecture in Dataset 1 [1].

	Test Accuracy			F1-Score			Params	Training Time (ms)	Inference Time (ms)
	Min	Max	Mean	Min	Max	Mean			
3 Block	0.882	0.936	0.906	0.865	0.928	0.893	1,437,153	41.95	0.07
4 Block	0.906	0.953	0.924	0.895	0.944	0.915	372,865	48.98	0.07
5 Block	**0.923**	**0.964**	**0.941**	**0.910**	**0.960**	**0.932**	**80,385**	**39.62**	**0.06**

Table 4.5: Comparison of different structures for CNN-based architecture in Dataset 1 [1].

Another dataset in which CNN-based architecture is used is Dataset 2 [2]. Since this dataset contains spectrograms from three different frequencies, two separate experiments were performed. The first of these has only 77GHz frequency spectrograms.

Table 4.6 shows experiments with different numbers of blocks using 77 GHz spectrograms.

As seen from Table 4.6, neural network models with more CNN blocks give better results and reduce the number of parameters. Therefore, using a model containing 5 CNN blocks is more advantageous for Dataset 2 [2].

	Test Accuracy			Params	Training Time (sec)	Inference Time (sec)
	Min	Max	Mean			
3 Block	**0.674**	0.713	0.691	544,475	**37.6**	**0.13**
4 Block	0.651	**0.746**	0.710	153,643	27.27	0.15
5 Block	0.666	0.731	**0.711**	**57,723**	35.63	0.16

Table 4.6: Comparison of different structures for CNN-based architecture in Dataset 2 [2] with only 77 GHz radar.

The second experiment with Dataset 2 [2] was on the model in which we used all frequencies (10 GHz, 24 GHz, and 77 GHz) in parallel, as shown in Table 4.7.

	Test Accuracy			Params	Training Time (ms)	Inference Time (ms)
	Min	Max	Mean			
2 Block	**0.819**	0.835	0.828	3,283,211	**50.62**	**0.06**
3 Block	0.817	0.842	**0.829**	3,290,603	70.26	0.07
4 Block	0.814	**0.890**	0.816	**959,435**	146.56	0.07

Table 4.7: Comparison of different structures for CNN-Based Architecture Dataset 2 [2] with all of the three radars.

In this experiment, as in previous experiments, while the number of blocks increases, the number of parameters decreases, but the test accuracy also decreases. Therefore, using three blocks in this architecture seems to be the most optimized solution for the number of parameters and test accuracy.

The decrease in test accuracy is because with the increasing number of parallel paths, more parameters are needed for better learning.

4.4.1.2 LSTM Based Activity Recognition

According to Table 4.8 and Table 4.5, we see an increase in test accuracy and F1 score values when we train Dataset 1 [1] with the LSTM-based model instead of the CNN-based model. We also observe that the number of parameters of the LSTM-based model is less in the models with few blocks. When we examine only Table 4.8, it is seen that increasing the number of blocks also increases the number of parameters. Also, the two-block LSTM-based model gives the best test accuracy and F1 score results.

	Test Accuracy			F1-Score			Params	Training Time (ms)	Inference Time (ms)
	Min	Max	Mean	Min	Max	Mean			
1 Block	0.941	0.965	0.955	0.930	**0.959**	0.940	**274,332**	**124.66**	**0.06**
2 Block	**0.947**	**0.965**	**0.957**	**0.940**	0.956	**0.950**	668,754	253.76	0.07
3 Block	0.929	0.947	0.939	0.909	0.938	0.924	1,063,293	243.42	0.07

Table 4.8: Comparison of different structures for LSTM-based architecture in Dataset 1 [1].

According to Table 4.9, in the LSTM-based model trained using only 77 GHz frequency in Dataset 2 [2], as the number of blocks increases, the number of parameters also increases, and the test accuracy performance of the model increases. In addition, training and inference times are getting longer. In this context, the user with no problem with the number of parameters, training, and inference times can use three blocks or more. However, one-block LSTM-based models can be used where there is the number of parameters, training, and inference time constraints.

	Test Accuracy			Params	Training Time (ms)	Inference Time (ms)
	Min	Max	Mean			
1 Block	0.796	0.818	0.805	**562,699**	**67.12**	**0.70**
2 Block	0.839	0.861	0.850	956,811	108.95	1.30
3 Block	**0.850**	**0.861**	**0.855**	1,350,987	117.99	1.79

Table 4.9: Comparison of different structures for LSTM-based architecture in Dataset 2 [2] with only 77 GHz radar.

When all frequencies are used in parallel in Dataset 2 [2] that can be seen in

Table 4.10, we can make a similar inference with Table 4.9. However, as a result of the test accuracy, the 2-block model looks better than the 3-block model. For this reason, we can say that the 2-block LSTM-based model is more advantageous than the three-block models when all frequencies are used in parallel.

	Test Accuracy			Params	Training Time (ms)	Inference Time (ms)
	Min	Max	Mean			
1 Block	0.883	0.930	0.912	**1,223,178**	**100.30**	**2.02**
2 Block	**0.919**	0.924	**0.922**	2,405,769	161.11	3.48
3 Block	0.910	**0.932**	0.917	3,588,426	200.77	7.54

Table 4.10: Comparison of different structures for LSTM-based architecture in Dataset 2 with all of the three radars.

4.4.1.3 LSTM- and GRU-based Activity Recognition

When we feed and test the LSTM- and GRU-based model with Dataset 1 [1], we see in Table 4.11 that the performance improved in both mean test accuracy and mean F1 score compared to the model with LSTM alone. Also, increasing the number of blocks increases the number of parameters and the training time and inference time, as expected. Mean test accuracy and F1 score values give the best results in the second block. Another significant output in Table 4.11 is that the maximum test accuracy value is 0.982, and the minimum value is 0.929 when using three blocks. From this, we understand that the standard deviation is also high, and the 3-block model does not give consistent results.

	Test Accuracy			F1-Score			Params	Training Time (ms)	Inference Time (ms)
	Min	Max	Mean	Min	Max	Mean			
1 Block	0.929	0.953	0.941	0.920	0.945	0.931	**484,87**	**283.06**	**0.20**
2 Block	**0.953**	**0.971**	**0.963**	**0.944**	0.966	**0.956**	1,273,715	490.17	0.22
3 Block	0.929	0.982	0.959	0.916	**0.980**	0.951	2,062,793	533.54	0.26

Table 4.11: Comparison of different structures for LSTM- and GRU-based architecture in Dataset 1 [1].

When analyzing Table 4.12 and Table 4.13, we can make the same inference as we did in the LSTM based model to determine the number of blocks in the

LSTM- and GRU-based model. When the LSTM- and GRU-based model is trained using only 77 GHz, we see that the test accuracy, number of parameters, training time, and inference time increase as the number of blocks increases. On the other hand, when three frequencies are fed in parallel, the increased number of blocks increases the number of parameters, training time, and inference time, but we see a decrease in test accuracy. Therefore, it is better to use a two-block model in this case.

	Test Accuracy			Params	Training Time (ms)	Inference Time (ms)
	Min	Max	Mean			
1 Block	0.821	0.847	0.837	**409,088**	**138.87**	**1.64**
2 Block	**0.864**	0.868	0.866	705,408	202.38	3.76
3 Block	0.861	**0.884**	**0.873**	1,001,856	265.50	4.40

Table 4.12: Comparison of different structures for LSTM- and GRU-based architecture in Dataset 2 with only 77 GHz radar.

	Test Accuracy			Params	Training Time (ms)	Inference Time (ms)
	Min	Max	Mean			
1 Block	0.925	0.938	0.929	**2,214,921**	**173.25**	**3.62**
2 Block	0.915	**0.946**	**0.939**	4,286,727	183.60	6.77
3 Block	**0.930**	0.941	0.936	6,358,665	267.56	10.18

Table 4.13: Comparison of different structures for LSTM- and GRU-based architecture in Dataset 2 with all of the three radars.

4.4.1.4 Siamese Multi-Frequency Activity Recognition

We created a Siamese-based model using the same structure as the LSTM- and GRU-based model but keeping the weights between parallel lines the same. With this model, the number of parameters is reduced since the same weights are used for each different path. When Table 4.13 and Table 4.14 are compared, it is seen that although the number of blocks is the same, the number of parameters has decreased by more than half thanks to Siamese-based Network. We also see that the mean test accuracy increases with the Siamese-based model. Since the

number of samples in the Dataset 2 [2] is not enough (60 samples for each of the 11 classes), we can increase the mean test accuracy by reducing the system's complexity.

When we examine the experimental results of the Siamese-based model in Table 4.14, we can observe that the increase in the number of blocks does not increase the mean test accuracy but increases the number of parameters, training, and inference time. Therefore, the one-block Siamese-based model is the best option in terms of test accuracy and the number of parameters.

	Test Accuracy			Params	Training Time (ms)	Inference Time (ms)
	Min	Max	Mean			
1 Block	**0.930**	0.952	**0.943**	**1,283,637**	**224.00**	**4.86**
2 Block	0.917	0.957	0.941	1,974,239	368.99	8.78
3 Block	0.930	**0.960**	0.940	2,666,079	405.46	13.29

Table 4.14: Comparison of different structures for Siamese-based architecture in Dataset 2.

4.4.2 Data Augmentation Methods

There are not enough examples in both Dataset 2 [2] and Dataset 1 [1] to train the models we propose. While there are 171 samples in total in Dataset 1 [1], there are only 60 samples for each class in the Dataset 2 [2]. Since the radar configurations of the datasets prepared using FMCW radar data are different, dissimilar datasets cannot be combined. Therefore, we need to use the Data Augmentation method to increase the number of data. We used two different Data Augmentation methods in our experiments. The first is MixUp Augmentation [109]. The second is the Window-Cropping method [46]. For all proposed methods, we compared three different methods in Table 4.15. The first is not to use the Data Augmentation method, the second is to use MixUp for Data Augmentation, and the third is to do Data Augmentation with Window-Cropping. While performing these operations, no changes were made within the blocks of

the model for the reliability of the comparisons. On the other hand, the Window-Cropping method has not been tested on CNN-based models because, by definition, the method cannot work correctly on CNN-based models. Furthermore, all results and code structure in Table 4.15 can be viewed on GitHub page at "https://github.com/mertege/Thesis_Experiments".

	Test Accuracy		
	Min	Max	Mean
Data Augmentation is not used with CNN-based in Dataset 1	0.782	0.900	0.839
Mix-up is used with CNN-based in Dataset 1	**0.923**	**0.964**	**0.941**
Data Augmentation is not used with CNN-based in 77 GHz	0.486	0.583	0.507
Mix-up is used with CNN-based in 77 GHz	**0.666**	**0.731**	**0.711**
Data Augmentation is not used with CNN-based in Dataset 2	0.743	0.819	0.797
Mix-up is used with CNN-based in Dataset 2	**0.817**	**0.842**	**0.829**
Data Augmentation is not used with LSTM-based in Dataset 1	0.909	0.956	0.928
Mix-up is used with LSTM-based in Dataset 1	**0.947**	**0.965**	**0.957**
Window Cropping is used with LSTM-based in Dataset 1	0.929	0.959	0.945
Data Augmentation is not used with LSTM-based in 77 GHz	**0.850**	**0.861**	**0.855**
Mix-up is used with LSTM-based in 77 GHz	0.843	0.852	0.849
Window Cropping is used with LSTM-based in 77 GHz	0.836	0.850	0.845
Data Augmentation is not used with LSTM-based in Dataset 2	**0.919**	0.924	0.922
Mix-up is used with LSTM-based in Dataset 2	0.913	0.930	0.922
Window Cropping is used with LSTM-based in Dataset 2	0.916	**0.932**	**0.923**
Data Augmentation is not used with LSTM- and GRU-based in Dataset 1	0.835	0.935	0.900
Mix-up is used with LSTM- and GRU-based in Dataset 1	**0.953**	**0.971**	**0.963**
Window Cropping is used with LSTM- and GRU-based in Dataset 1	0.953	0.965	0.959
Data Augmentation is not used with LSTM- and GRU-based in 77 GHz	0.861	0.884	0.873
Mix-up is used with LSTM- and GRU-based in 77 GHz	0.858	0.880	0.866
Window Cropping is used with LSTM- and GRU-based in 77 GHz	**0.861**	**0.886**	**0.876**
Data Augmentation is not used with LSTM- and GRU-based in Dataset 2	0.925	0.938	0.929
Mix-up is used with LSTM- and GRU-based in Dataset 2	0.910	0.929	0.918
Window Cropping is used with LSTM- and GRU-based in Dataset 2	**0.930**	**0.957**	**0.944**
Data Augmentation is not used with Siamese-based in Dataset 2	**0.930**	**0.952**	**0.943**
Mix-up is used with Siamese-based in Dataset 2	0.930	0.943	0.937
Window Cropping is used with Siamese-based in Dataset 2	0.911	0.944	0.928

Table 4.15: Data Augmentation methods, including Mix-Up and Window-Cropping, are compared. All proposed methods are experimented with No Augmentation, Mix-Up, and Window-Cropping methods.

When the CNN-based models in Table 4.15 are examined, we observe that the MixUp Augmentation method increases the mean test accuracy value in all three datasets. Since MixUp is an image-oriented Data Augmentation method, the result in Table 4.15 shows that MixUp works well for CNN-based models.

On the other hand, when we examine the effect of Data Augmentation methods, including MixUp and Window-Cropping, on models containing time series structures, we see a performance increase in models containing only Dataset 1 [1]. Dataset 2 [2] does not provide a significant increase in test accuracy. This is because Dataset 1 [1] consists of only two classes, while Dataset 2 [2] consists of 11 different classes. Also, the feature difference between classes in Dataset 2 [2] is not much because there is not much difference between "bending", "kneel" and "pick up an object" classes in terms of FMCW radar data. For this reason, the applied Data Augmentation methods could not reveal the difference between the properties of these classes.

4.4.3 Ablation Study

By performing an ablation study, we demonstrate the effect of Attention and Global Average Pooling, which we use in the proposed models, on test accuracy performance. The Attention mechanism, the working process described in Section 3.6, gives greater coefficient values to the loss-reducing features during training. On the other hand, Global Average Pooling is implemented with 2D input passing through CNN, LSTM, or GRU blocks before feeding the dense layer. In this way, 2D input can be converted to 1D. Global Average Pooling sums all time steps along the time axis, and the depth of the 1D dimension is reduced to the number of features. Flattening was used instead of Global Average Pooling to convert 2D images to 1D while performing the ablation study.

4.4.3.1 Attention Mechanism

The Ablation Study was applied to 2 models using two different datasets for Attention Mechanism. The first is the Siamesed-based network using Dataset 2 [2], and the second is the LSTM- and GRU-based model using Dataset 1 [1].

When the Attention model is removed for both models, it is seen in Table 4.16 that the test accuracy results drop too much. There is a **%3** increase in mean test

	Test Accuracy		
	Min	Max	Mean
Siamesed-based Network with no Attention	0.913	0.929	0.913
Siamesed-based Network with Attention	**0.930**	**0.952**	**0.943**
LSTM- and GRU-based in Dataset 1 with no Attention	0.912	0.953	0.933
LSTM- and GRU-based in Dataset 1 with Attention	**0.953**	**0.971**	**0.963**

Table 4.16: An ablation study was performed to analyze the effect of Attention block on test accuracy results. Siamese-based and LSTM- and GRU-based models are used for the ablation of Attention block.

accuracy for both models. The high positive effect of the Attention block for both models demonstrates the importance of this block for time series classification.

4.4.3.2 Global Averaging Pool

Parameter heading is also added to the comparison chart when performing the Ablation Study for Global Mean Pooling. This is because, as can be seen from Table 4.17, when Flattening is used instead of Global Average Pooling, the number of parameters increases dramatically because, in Global Average Pooling, all time steps for each feature are summed together to get a single value. Otherwise, the number of parameters increases too much because the Flatten layer flattens all time steps and uses it for the rest of the model.

On the other hand, when Global Average Pooling is used for both models, it is seen in Table 4.17 that the mean test accuracy values increase. This is because possible overfitting is prevented by decreasing the number of parameters.

	Test Accuracy			Parameters
	Min	Max	Mean	
Siamesed-based Network with Flatten	0.927	0.948	0.935	35,412,725
Siamesed-based Network with Global Pooling	**0.930**	**0.952**	**0.943**	**1,283,637**
LSTM- and GRU-based in Dataset 1 with Flatten	0.947	0.965	0.955	18,060,355
LSTM- and GRU-based in Dataset 1 with Global Pooling	**0.953**	**0.971**	**0.963**	**1,273,715**

Table 4.17: An ablation study was performed to analyze the effect of the Global Average Pooling block on test accuracy results. Siamese-based and LSTM- and GRU-based models are used for the ablation of the Global Average Pooling.

4.4.4 Dataset 1 [1]

In this part, we compare the proposed models using Dataset 1 [1]. Range Doppler and Spectrogram data are fed in parallel with the proposed models (CNN-based, LSTM-based, and LSTM- and GRU-based). We feed all models with Range Doppler or Spectrogram only to show the effect on the test accuracy of feeding two different inputs together. Table 4.18 shows that all proposed models were tested with three different inputs (Spectrogram, Range Doppler, Spectrogram and Range Doppler).

	Test Accuracy			F1-Score			Params	Training Time (ms)	Inference Time (ms)
	Min	Max	Mean	Min	Max	Mean			
Spectrogram CNN-based	0.853	0.912	0.898	0.844	0.902	0.885	**39,41**	**16.25**	0.06
Range-Doppler CNN-based	0.888	0.929	0.918	0.876	0.919	0.906	45,553	16.536	0.06
Spec-RD CNN-based	0.923	0.964	0.941	0.910	0.960	0.932	80,385	39.62	**0.06**
Spectrogram LSTM-based	0.894	0.918	0.908	0.869	0.899	0.885	75,337	42.713	0.16
Range-Doppler LSTM-based	0.935	0.982	0.955	0.926	0.980	0.948	77,049	63.520	0.08
Spec-RD LSTM-based	0.947	0.965	0.957	0.940	0.956	0.950	668,754	253.761	0.07
Spectrogram LSTM- and GRU-based	0.882	0.912	0.896	0.860	0.895	0.877	226,590	121.990	0.26
Range-Doppler LSTM- and GRU-based	0.947	0.965	0.953	0.939	0.959	0.947	225,697	113.621	0.19
Spec-RD LSTM- and GRU-based	**0.953**	**0.971**	**0.963**	**0.944**	**0.966**	**0.956**	1,273,715	490.175	0.22
Baseline Method 1 [86]	-	-	0.935	-	-	-	-	-	-

Table 4.18: Comparison of proposed methods and Baseline Method 1 [86] for Dataset 1 [1].

The mean test accuracy is the lowest in all three models when only the Spectrogram is used as input in Table 4.18. Also, when we use Spectrogram and Range Doppler together, we get the best test accuracy results for all models. Since Range Doppler and Spectrogram plots represent different features from radar data, using both together is expected to increase model accuracy and F1-Score. Table 4.18 supports this thesis. On the other hand, because we use two different inputs, our model has two different paths. This second path costs us extra parameter count, training, and inference time. Table 4.18 shows the increase in the number of parameters, training, and inference times.

Baseline Method 1 [86] tries to classify slow, and fast human movement as we do, using the Dataset 1 [1] we use. Since the authors of [86] only share mean test accuracy, other headings in the row of Baseline Method 1 [86] in Table 4.18 are left blank. When we compare the mean test accuracy values, we observe that all the proposed models surpass the results of Baseline Method 1 [86] when we

use both Range-Doppler and Spectrogram together as inputs. In addition, when only Range Doppler input is used in LSTM-based and LSTM- and GRU-based models, it gives better results than Baseline Method 1 [86].

4.4.5 Dataset 2 [2]

In this section, experiments are carried out on the proposed models using Dataset 2 [2]. Since this dataset contains Spectrograms with three different center frequencies (10 GHz, 24 GHz, and 77 GHz), the proposed models are tested separately for each frequency. In addition, experiments were carried out using three frequencies together. As a result of the experiments, test accuracy values (min, max, and mean), number of parameters, training, and inference time are included in the comparison titles.

CNN-based, LSTM-based, and LSTM- and GRU-based models were tested for each frequency used separately and when all frequencies were used simultaneously. In addition to these models, the Siamese-based model has been tested only in the multi-frequency condition, as the architecture of the Siamese-based model needs three inputs in parallel.

4.4.5.1 One-Input

There are three different radars in Dataset 2 [2], and each has a different center frequency. This section tests data from each radar separately on all proposed models. In this way, the performance of our models is measured with various radars.

When we test the models with radar data operating at 10 GHz, we observe that we get the best test accuracy in the architecture where LSTM and GRU work in parallel. The model using only LSTM ranks second in test accuracy performance, as shown in Table 4.19. On the other hand, as the test accuracy value increases, the number of parameters, training, and inference times also increase.

	Test Accuracy			Params	Training Time (ms)	Inference Time (ms)
	Min	Max	Mean			
CNN-based Arch	0.713	0.754	0.732	**57,723**	**47.72**	**0.31**
LSTM-based	0.844	0.861	0.849	1,350,987	123.86	1.90
LSTM- and GRU-based Arch	**0.865**	**0.887**	**0.876**	2,370,187	260.82	4.53

Table 4.19: Comparison of different methods and Baseline Method 2 for 10 GHz in Dataset 2 [2].

	Test Accuracy			Params	Training Time (ms)	Inference Time (ms)
	Min	Max	Mean			
CNN-based Arch	0.733	0.753	0.741	**57,723**	**92.041**	**0.309**
LSTM-based Arch	0.836	0.849	0.844	1,350,987	760.70	2.25
LSTM- and GRU-based Arch	**0.842**	**0.855**	**0.850**	2,370,187	322.539	3.336

Table 4.20: Comparison of different methods and Baseline Method 2 for 24 GHz in Dataset 2 [2].

When 24 GHz and 77 GHz center frequency radars are used to test our models (Table 4.20 and Table 4.21), we make inferences similar to radar using 10 GHz frequencies.

	Test Accuracy			Params	Training Time (ms)	Inference Time (ms)
	Min	Max	Mean			
CNN-based Arch	0.666	0.731	0.711	**57,723**	**35.63**	**0.16**
LSTM-based Arch	0.850	0.861	0.855	1,350,987	117.99	1.79
LSTM- and GRU-based Arch	**0.861**	**0.884**	**0.873**	1,001,856	265.50	4.40

Table 4.21: Comparison of different methods and Baseline Method 2 for 77 GHz in Dataset 2 [2].

Since we get similar results from different radars, we can say that using the LSTM-based model is a better option for time series classification using a CNN-based model. Also, when LSTM and GRU are used in parallel, it gives better test accuracy than LSTM alone. When we examine the comparison of the models in terms of the number of parameters, training, and inference times, we can see that the CNN-based model is the most advantageous. In this context, the LSTM-based model takes second place.

4.4.5.2 Multi-Input

Comparing Table 4.21 and 4.22, we can see that each model gives better results if we feed the outputs of all three radars in parallel instead of feeding the proposed models with a single input. Although this model change increases the test accuracy value, it also increases the number of parameters since each input path uses different parameters.

	Test Accuracy			Params	Training Time (ms)	Inference Time (ms)
	Min	Max	Mean			
CNN-based Arch	0.817	0.842	0.829	**290,603**	**70.26**	**0.07**
LSTM-Based Arch	0.919	0.924	0.922	2,405,769	161.12	3.48
LSTM- and GRU-based Arch	0.915	0.946	0.939	4,286,727	183.60	6.77
Siamese-based Arch	**0.930**	**0.952**	**0.943**	1,283,637	224.00	4.86
Baseline Model 2 [2]	-	-	0.915	-	-	-

Table 4.22: Comparison of different methods and Baseline Method 2 for Multi-Frequency in Dataset 2 [2].

Since the Siamese-based model is designed for the multiple-entry situation, this model can be tested in only the multi-input case. Since all weights passing through parallel paths in Siamese-based architecture are tied, it reduces the number of parameters by at least one-third, shown in Table 4.22. Additionally, our mean test accuracy increases by four percent as the probability of overfitting decreases.

The bottom row of table 4.22 shows the mean test accuracy result of [2]. This model is defined as the Baseline Model 2 because Dataset 2 is taken from this article. Authors of [2] used the Convolutional Autoencoder (CAE) and trained their models by taking data from a single radar instead of a multiple input option. Therefore, all our models, except the CNN-based model, outperformed Baseline Model 2.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we propose four different models to classify human movement. These models are CNN-based, LSTM-based, LSTM- and GRU-based, and Siamese-based. As far as we know, the LSTM- and GRU-based model is the first where the input is fed in parallel to the LSTM and GRU blocks. To increase the reliability of our proposed models, we conducted our experiments on two different datasets, Dataset 1 [1] and Dataset 2 [2]. If we add the GRU path in parallel to a model that we have classified using LSTM only, we observe an increase in all test accuracy results in the experiments.

Furthermore, the Siamese-based model is, to our best knowledge, the first study in which input data from three different radars in Dataset 2 is fed in parallel and tied weights. When we feed CNN-based, LSTM-based, and LSTM- and GRU-based models as multiple inputs, test accuracy results increase by an average of 7%. Also, if we use tied weights between paths in the Siamese-based model, we reduce the number of parameters by one quarter and increase the test accuracy result by 0.4%.

We used two "Data Augmentation" methods to feed the models we created with more data. These are Mix-Up [45] and Window Cropping [46]. Test accuracy increased when the proposed models were tested in Dataset 1 with these data augmentation methods. On the other hand, when the proposed models are tested in Dataset 2, no positive effect of Data Augmentation methods on test accuracy is observed. This is because 11 classes in Dataset 2 are very close to each other in action, such as "bending", "kneel", and "pick up an object". So, the data augmentation method cannot increase the feature difference between classes.

Besides, the Ablation Study of the Attention Mechanism and Global Average Pooling blocks was conducted. The effects of these blocks on classifying human movements were observed. Attention Mechanism and Global Average Pooling provide a 3% and 1% increase in test accuracy, respectively. In addition, Global Average Pooling significantly reduces the number of parameters.

5.2 Future Work

Multi-input models were tested with Dataset 2 [2]. To show the positive effect of the multi-input model on test accuracy performance, we can test our models with more datasets. However, the number of datasets using different frequency radars scanning the same scene is not as large as other radar datasets. For this reason, a dataset containing the multi-radar condition can be created.

We also need to reduce inference times so that Human Activity Recognition models can work in real-time. For this, we can apply quantization methods to these models. Thus, both the number of parameters and the inference times can be reduced.

Bibliography

[1] S. Björklund, H. Petersson, and G. Hendeby, "Features for micro-doppler based activity classification," *IET Radar, Sonar & Navigation*, vol. 9, no. 9, pp. 1181–1187, 2015.

[2] S. Z. Gurbuz, M. M. Rahman, E. Kurtoglu, T. Macks, and F. Fioranelli, "Cross-frequency training with adversarial learning for radar micro-doppler signature classification (rising researcher)," in *Radar Sensor Technology XXIV*, vol. 11408, p. 114080A, International Society for Optics and Photonics, 2020.

[3] J. V. Tembhurne and T. Diwan, "Sentiment analysis in textual, visual and multimodal inputs using recurrent neural networks," *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 6871–6910, 2021.

[4] A. Palffy, J. Dong, J. F. Kooij, and D. M. Gavrila, "Cnn based road user detection using the 3d radar cube," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1263–1270, 2020.

[5] D. Cha, S. Jeong, M. Yoo, J. Oh, and D. Han, "Multi-input deep learning based fmcw radar signal classification," *Electronics*, vol. 10, no. 10, p. 1144, 2021.

[6] W.-L. Lu and J. J. Little, "Simultaneous tracking and action recognition using the pca-hog descriptor," in *The 3rd Canadian Conference on Computer and Robot Vision (CRV'06)*, pp. 6–6, IEEE, 2006.

[7] Y. Ke, R. Sukthankar, and M. Hebert, "Spatio-temporal shape and flow correlation for action recognition," in *2007 IEEE conference on computer vision and pattern recognition*, pp. 1–8, IEEE, 2007.

[8] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden markov model.," in *CVPR*, vol. 92, pp. 379–385, 1992.

[9] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 12, pp. 2247–2253, 2007.

[10] Y. Du, F. Chen, and W. Xu, "Human interaction representation and recognition through motion decomposition," *IEEE Signal Processing Letters*, vol. 14, no. 12, pp. 952–955, 2007.

[11] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *2005 IEEE international workshop on visual surveillance and performance evaluation of tracking and surveillance*, pp. 65–72, IEEE, 2005.

[12] C. A. Ronao and S.-B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert systems with applications*, vol. 59, pp. 235–244, 2016.

[13] Y.-M. Kuo, J.-S. Lee, and P.-C. Chung, "A visual context-awareness-based sleeping-respiration measurement system," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 2, pp. 255–265, 2009.

[14] R. Bodor, B. Jackson, and N. Papanikolopoulos, "Vision-based human tracking and activity recognition," in *Proc. of the 11th Mediterranean Conf. on Control and Automation*, vol. 1, pp. 1–6, Citeseer, 2003.

[15] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden markov models for complex action recognition," in *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, pp. 994–999, IEEE, 1997.

[16] S. Kazmi and I. J. Palmer, "Action recognition for support of adaptive gameplay: A case study of a first person shooter," *International Journal of Computer Games Technology*, vol. 2010, 2010.

[17] M. Z. Uddin and A. Soylu, "Human activity recognition using wearable sensors, discriminant analysis, and long short-term memory-based neural structured learning," *Scientific Reports*, vol. 11, no. 1, pp. 1–15, 2021.

[18] J. Shao, K. Kang, C. Change Loy, and X. Wang, "Deeply learned attributes for crowded scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4657–4666, 2015.

[19] J. S. Patel, F. Fioranelli, and D. Anderson, "Review of radar classification and rcs characterisation techniques for small uavs or drones," *IET Radar, Sonar & Navigation*, vol. 12, no. 9, pp. 911–919, 2018.

[20] A. Zyweck and R. E. Bogner, "Radar target classification of commercial aircraft," *IEEE Transactions on Aerospace and Electronic systems*, vol. 32, no. 2, pp. 598–606, 1996.

[21] A. Profeta, A. Rodriguez, and H. S. Clouse, "Convolutional neural networks for synthetic aperture radar classification," in *Algorithms for synthetic aperture radar imagery XXIII*, vol. 9843, pp. 185–194, SPIE, 2016.

[22] P. E. Zwicke and I. Kiss, "A new implementation of the mellin transform and its application to radar classification of ships," *IEEE Transactions on pattern analysis and machine intelligence*, no. 2, pp. 191–199, 1983.

[23] T. Gindele, S. Brechtel, J. Schroder, and R. Dillmann, "Bayesian occupancy grid filter for dynamic environments using prior map knowledge," in *2009 IEEE Intelligent Vehicles Symposium*, pp. 669–676, IEEE, 2009.

[24] S. Wirges, T. Fischer, C. Stiller, and J. B. Frias, "Object detection and classification in occupancy grid maps using deep convolutional networks," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 3530–3535, IEEE, 2018.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[26] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[27] B. Major, D. Fontijne, A. Ansari, R. Teja Sukhavasi, R. Gowaikar, M. Hamilton, S. Lee, S. Grzechnik, and S. Subramanian, "Vehicle detection with automotive radar using deep learning on range-azimuth-doppler tensors," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

[28] P. Molchanov, R. I. Harmanny, J. J. de Wit, K. Egiazarian, and J. Astola, "Classification of small uavs and birds by micro-doppler signatures," *International Journal of Microwave and Wireless Technologies*, vol. 6, no. 3-4, pp. 435–444, 2014.

[29] L. Mo, F. Li, Y. Zhu, and A. Huang, "Human physical activity recognition based on computer vision with deep learning model," in *2016 IEEE international instrumentation and measurement technology conference proceedings*, pp. 1–6, IEEE, 2016.

[30] D. R. Beddiar, B. Nini, M. Sabokrou, and A. Hadid, "Vision-based human activity recognition: a survey," *Multimedia Tools and Applications*, vol. 79, no. 41, pp. 30509–30555, 2020.

[31] O. Halabi, S. Fawal, E. Almughani, and L. Al-Homsi, "Driver activity recognition in virtual reality driving simulation," in *2017 8th International Conference on Information and Communication Systems (ICICS)*, pp. 111–115, IEEE, 2017.

[32] C. Dhiman and D. K. Vishwakarma, "A review of state-of-the-art techniques for abnormal human activity recognition," *Engineering Applications of Artificial Intelligence*, vol. 77, pp. 21–45, 2019.

[33] M. Vrigkas, C. Nikou, and I. A. Kakadiaris, "A review of human activity recognition methods," *Frontiers in Robotics and AI*, vol. 2, p. 28, 2015.

[34] K. Kim, A. Jalal, and M. Mahmood, "Vision-based human activity recognition system using depth silhouettes: A smart home system for monitoring the residents," *Journal of Electrical Engineering & Technology*, vol. 14, no. 6, pp. 2567–2573, 2019.

[35] B. Vandersmissen, N. Knudde, A. Jalalvand, I. Couckuyt, A. Bourdoux, W. De Neve, and T. Dhaene, "Indoor person identification using a low-power fmcw radar," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 7, pp. 3941–3952, 2018.

[36] F. Luo, S. Poslad, and E. Bodanese, "Human activity detection and coarse localization outdoors using micro-doppler signatures," *IEEE Sensors Journal*, vol. 19, no. 18, pp. 8079–8094, 2019.

[37] X. Bai, Y. Hui, L. Wang, and F. Zhou, "Radar-based human gait recognition using dual-channel deep convolutional neural network," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 12, pp. 9767–9778, 2019.

[38] Y. Kim and H. Ling, "Human activity classification based on micro-doppler signatures using a support vector machine," *IEEE transactions on geoscience and remote sensing*, vol. 47, no. 5, pp. 1328–1337, 2009.

[39] S. Gupta, P. K. Rai, A. Kumar, P. K. Yalavarthy, and L. R. Cenkeramaddi, "Target classification by mmwave fmcw radars using machine learning on range-angle images," *IEEE Sensors Journal*, vol. 21, no. 18, pp. 19993–20001, 2021.

[40] O. Postolache, P. S. Girão, R. N. Madeira, and G. Postolache, "Microwave fmcw doppler radar implementation for in-house pervasive health care system," in *2010 IEEE International Workshop on Medical Measurements and Applications*, pp. 47–52, IEEE, 2010.

[41] K. Stasiak, M. Ciesielski, M. Khyzhniak, K. Jedrzejewski, J. Kłos, A. Droszcz, and S. Brawata, "Fmcw perimeter radar for border protection-preliminary field trials," in *2020 21st International Radar Symposium (IRS)*, pp. 11–14, IEEE, 2020.

[42] E. Gambi, G. Ciattaglia, and A. De Santis, "People movement analysis with automotive radar," in *2019 IEEE 23rd International Symposium on Consumer Technologies (ISCT)*, pp. 233–237, IEEE, 2019.

[43] R. Ricci and A. Balleri, "Recognition of humans based on radar micro-doppler shape spectrum features," *IET Radar, Sonar & Navigation*, vol. 9, no. 9, pp. 1216–1223, 2015.

[44] F. Fioranelli, M. Ritchie, and H. Griffiths, "Analysis of polarimetric multistatic human micro-doppler classification of armed/unarmed personnel," in *2015 IEEE Radar Conference (RadarCon)*, pp. 0432–0437, 2015.

[45] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[46] Z. Cui, W. Chen, and Y. Chen, "Multi-scale convolutional neural networks for time series classification," *arXiv preprint arXiv:1603.06995*, 2016.

[47] J. Cao, Z. Li, and J. Li, "Financial time series forecasting model based on ceemdan and lstm," *Physica A: Statistical mechanics and its applications*, vol. 519, pp. 127–139, 2019.

[48] P. Esling and C. Agon, "Multiobjective time series matching for audio classification and retrieval," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 2057–2072, 2013.

[49] J. Lin, E. Keogh, A. Fu, and H. Van Herle, "Approximations to magic: Finding unusual medical time series," in *18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05)*, pp. 329–334, IEEE, 2005.

[50] L. Eklundh and P. Jönsson, "Timesat for processing time-series data from satellite sensors for land surface monitoring," in *Multitemporal Remote Sensing*, pp. 177–194, Springer, 2016.

[51] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[52] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, 2017.

[53] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and information systems*, vol. 7, no. 3, pp. 358–386, 2005.

[54] G. Saggio, P. Cavallo, M. Ricci, V. Errico, J. Zea, and M. E. Benalcázar, "Sign language recognition using wearable electronics: implementing k-nearest neighbors with dynamic time warping and convolutional neural network algorithms," *Sensors*, vol. 20, no. 14, p. 3879, 2020.

[55] T. M. Tran, X.-M. T. Le, H. T. Nguyen, and V.-N. Huynh, "A novel non-parametric method for time series classification based on k-nearest neighbors and dynamic time warping barycenter averaging," *Engineering Applications of Artificial Intelligence*, vol. 78, pp. 173–185, 2019.

[56] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[57] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *ieee Computational intelligenCe magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[58] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 8599–8603, IEEE, 2013.

[59] J. Kiefer and J. Wolfowitz, "Stochastic estimation of the maximum of a regression function," *The Annals of Mathematical Statistics*, pp. 462–466, 1952.

[60] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[61] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[62] E. Buber and D. Banu, "Performance analysis and cpu vs gpu comparison for deep learning," in *2018 6th International Conference on Control Engineering & Information Technology (CEIT)*, pp. 1–6, IEEE, 2018.

[63] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[64] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[65] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[66] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[67] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[68] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina, *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4774–4778, IEEE, 2018.

[69] M. R. Costa-jussà, J. Cross, O. Çelebi, M. Elbayad, K. Heafield, K. Heffernan, E. Kalbassi, J. Lam, D. Licht, J. Maillard, *et al.*, "No language left behind: Scaling human-centered machine translation," *arXiv e-prints*, pp. arXiv–2207, 2022.

[70] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[71] J. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[72] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[73] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International joint conference on neural networks (IJCNN)*, pp. 1578–1585, IEEE, 2017.

[74] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "Lstm fully convolutional networks for time series classification," *IEEE access*, vol. 6, pp. 1662–1669, 2017.

[75] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 95–104, 2018.

[76] N. Wu, B. Green, X. Ben, and S. O'Banion, "Deep transformer models for time series forecasting: The influenza prevalence case," *arXiv preprint arXiv:2001.08317*, 2020.

[77] S.-Y. Shih, F.-K. Sun, and H.-y. Lee, "Temporal pattern attention for multivariate time series forecasting," *Machine Learning*, vol. 108, no. 8, pp. 1421–1441, 2019.

[78] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[79] M. Liu, S. Ren, S. Ma, J. Jiao, Y. Chen, Z. Wang, and W. Song, "Gated transformer networks for multivariate time series classification," *arXiv preprint arXiv:2103.14438*, 2021.

[80] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[81] L. Cai, J. Gao, and D. Zhao, "A review of the application of deep learning in medical image classification and segmentation," *Annals of translational medicine*, vol. 8, no. 11, 2020.

[82] G. Wang, J. C. Ye, and B. De Man, "Deep learning for tomographic image reconstruction," *Nature Machine Intelligence*, vol. 2, no. 12, pp. 737–748, 2020.

[83] J. M. Garcia, R. Prophet, J. C. F. Michel, R. Ebelt, M. Vossiek, and I. Weber, "Identification of ghost moving detections in automotive scenarios with deep learning," in *2019 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pp. 1–4, IEEE, 2019.

[84] A. Zhang, F. Nowruzi, and R. Laganiere, "Raddet: Range-azimuth-doppler based radar object detection for dynamic road users," in *2021 18th Conference on Robots and Vision (CRV)*, (Los Alamitos, CA, USA), pp. 95–102, IEEE Computer Society, may 2021.

[85] K. Patel, K. Rambach, T. Visentin, D. Rusev, M. Pfeiffer, and B. Yang, "Deep learning-based object classification on automotive radar spectra," in *2019 IEEE Radar Conference (RadarConf)*, pp. 1–6, IEEE, 2019.

[86] L. Senigagliesi, G. Ciattaglia, A. D. Santis, and E. Gambi, "People walking classification using automotive radar," *Electronics (Switzerland)*, vol. 9, 2020.

[87] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[88] H. Rohling, "Ordered statistic cfar technique-an overview," in *2011 12th International Radar Symposium (IRS)*, pp. 631–638, IEEE, 2011.

[89] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[90] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *Advances in neural information processing systems*, vol. 28, 2015.

[91] N. Ballas, L. Yao, C. Pal, and A. Courville, "Delving deeper into convolutional networks for learning video representations," *arXiv preprint arXiv:1511.06432*, 2015.

[92] A. Shrestha, H. Li, J. Le Kernec, and F. Fioranelli, "Continuous human activity classification from fmcw radar with bi-lstm networks," *IEEE Sensors Journal*, vol. 20, no. 22, pp. 13607–13619, 2020.

[93] S. Kim, S. Lee, S. Doo, and B. Shim, "Moving target classification in automotive radar systems using convolutional recurrent neural networks," in *2018 26th European Signal Processing Conference (EUSIPCO)*, pp. 1482–1486, IEEE, 2018.

[94] H. Abedi, A. Ansariyan, P. P. Morita, J. Boger, A. Wong, and G. Shaker, "Sequential deep learning for in-home activity monitoring using mm-wave fmcw radar," in *2021 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting (APS/URSI)*, pp. 1499–1500, IEEE, 2021.

[95] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[96] M. Mostajabi, C. M. Wang, D. Ranjan, and G. Hsyu, "High-resolution radar dataset for semi-supervised learning of dynamic objects," in *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 100–101, 2020.

[97] E. Gambi, G. Ciattaglia, A. D. Santis, and L. Senigagliesi, "Millimeter wave radar data of people walking," *Data in Brief*, vol. 31, p. 105996, 8 2020.

[98] T. T. Um, F. M. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić, "Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks," in *Proceedings of the 19th ACM international conference on multimodal interaction*, pp. 216–220, 2017.

[99] D. S. Park, Y. Zhang, C.-C. Chiu, Y. Chen, B. Li, W. Chan, Q. V. Le, and Y. Wu, "Specaugment on large scale datasets," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6879–6883, IEEE, 2020.

[100] T. DeVries and G. W. Taylor, "Dataset augmentation in feature space," *arXiv preprint arXiv:1702.05538*, 2017.

[101] T.-H. Cheung and D.-Y. Yeung, "Modals: Modality-agnostic automated data augmentation in the latent space," in *International Conference on Learning Representations*, 2020.

[102] J. Yoon, D. Jarrett, and M. Van der Schaar, "Time-series generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[103] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional gans," *arXiv preprint arXiv:1706.02633*, 2017.

[104] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 702–703, 2020.

[105] J. Li, Q.-F. Wang, R. Zhang, and K. Huang, "Mix-up augmentation for oracle character recognition with imbalanced data distribution," in *International Conference on Document Analysis and Recognition*, pp. 237–251, Springer, 2021.

[106] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[107] Q. Zheng, L. Yang, Y. Xie, J. Li, T. Hu, J. Zhu, C. Song, and Z. Xu, "A target detection scheme with decreased complexity and enhanced performance for range-doppler fmcw radar," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, 2021.

[108] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 international interdisciplinary PhD workshop (IIPhDW)*, pp. 117–122, IEEE, 2018.

[109] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018.

[110] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, pp. 448–456, PMLR, 2015.

[111] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[112] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *CoRR*, vol. abs/1505.00853, 2015.

[113] C. C. Aggarwal *et al.*, "Neural networks and deep learning," *Springer*, vol. 10, pp. 140–141, 2018.

[114] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, 2007.

[115] R. Bonnin, "Machine learning for developers. [electronic resource]," 2017.

[116] A. Dayal, N. Paluru, L. R. Cenkeramaddi, P. K. Yalavarthy, *et al.*, "Design and implementation of deep learning based contactless authentication system using hand gestures," *Electronics*, vol. 10, no. 2, p. 182, 2021.

[117] I. Kandel and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," *ICT express*, vol. 6, no. 4, pp. 312–315, 2020.

[118] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[119] Q. Gu, L. Zhu, and Z. Cai, "Evaluation measures of the classification performance of imbalanced data sets," in *International symposium on intelligence computation and applications*, pp. 461–471, Springer, 2009.

[120] D. K. Dewangan and S. P. Sahu, "Deep learning-based speed bump detection model for intelligent vehicle system using raspberry pi," *IEEE sensors journal*, vol. 21, no. 3, pp. 3570–3578, 2020.

[121] A. Shawahna, S. M. Sait, and A. El-Maleh, "Fpga-based accelerators of deep learning networks for learning and classification: A review," *ieee Access*, vol. 7, pp. 7823–7859, 2018.

[122] Y. Deng, "Deep learning on mobile devices: a review," in *Mobile Multimedia/Image Processing, Security, and Applications 2019*, vol. 10993, pp. 52–66, SPIE, 2019.