Towards Robust and Deployable Gesture and Activity Recognisers

Utkarsh Kunwar

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 31.10.2021

Supervisor

Prof. Antti Oulasvirta

Advisors

Dr. Ilhan Aslan

Dr. Juho Kannala

Aalto University
School of Electrical
Engineering

Aalto University
School of Electrical
Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Utkarsh Kunwar

Title Towards Robust and Deployable Gesture and Activity Recognisers

Degree programme ICT Innovation - EIT Digital Master School

Major Autonomous Systems **Code of major** ELEC3055

Supervisor Prof. Antti Oulasvirta

Advisors Dr. Ilhan Aslan, Dr. Juho Kannala

Date 31.10.2021 **Number of pages** 62+2 **Language** English

Abstract

Smartphones and wearables have become an extension of one's self, with gestures providing quick access to command execution, and activity tracking helping users log their daily life. Recent research in gesture recognition points towards common events like a user re-wearing or readjusting their smartwatch deteriorate recognition accuracy significantly. Further, the available state-of-the-art deep learning models for gesture or activity recognition are too large and computationally heavy to be deployed and run continuously in the background. This problem of engineering robust yet deployable gesture recognisers for use in wearables is open-ended. This thesis provides a review of known approaches in machine learning and human activity recognition (HAR) for addressing model robustness. This thesis also proposes variations of convolution based models for use with raw or spectrogram sensor data. Finally, a cross-validation based evaluation approach for quantifying individual and situational-variabilities is used to demonstrate that with an application-oriented design, models can be made two orders of magnitude smaller while improving on both recognition accuracy and robustness.

Keywords gesture recognition, deep learning, wearables, robustness

Preface

I would like to thank Professor Antti Oulasvirta and Professor Juho Kannala for their guidance and help me point towards new and exciting directions. Professor Antti's insight, advice, and freedom of thought has allowed me to explore several paradigms in this thesis. I want to thank Dr Ilhan Aslan for his help in broadening my gaze on certain topics while maintaining a result-oriented mindset for the company. I would also like to thank my colleagues Moritz Berghofer, Sheetal Borar, Antti Parviainen, Heidi Hassinen, and Julia Kylmälä for supporting me with the technical aspects of the project and for the constructive criticism.

Further, I want to thank my friends Dhruv Patel, Sri Hari Nemani, Vetcha Sai Subba Rao, and Dr Aparna Singh for helping me gather knowledge on the difficult topics in their respective domains, and for their unwavering support and unconditional love throughout the pandemic. But most importantly, I thank my parents Dr P. Kunwar and Neelam Kunwar for motivating me to work towards a bright future and for always keeping the flame of curiosity in me alight.

Espoo, 22.11.2021

Utkarsh Kunwar

Contents

Symbols and abbreviations

Symbols

$\mathbf{x_t}$	Input sample at time t
$\mathbf{h_t}$	Hidden state of RNN or LSTM
$\mathbf{y_t}$	Output at time t
W, U	Weight matrices
b	Bias vector
σ	Non-linearity or sigmoid function
$I_{i,j,k}$	Input image batch tensor
$K_{i,j,k}$	CNN kernel tensor
$\bar{x}$	Mean of variable x
$U(a, b)$	Uniform distribution between $[a, b)$
$\mathcal{N}(\mu, \sigma)$	Normal distribution with mean μ and standard deviation σ

Operators

$\sum_i$	Sum over index i
$\mathbf{x} \cdot \mathbf{y}$	Dot product of vectors $\mathbf{x}$ and $\mathbf{y}$

Abbreviations

AHRS	Attitude and Heading Reference System
BCE	Binary Cross Entropy
CNN	Convolutional Neural Network
CSV	Comma-Separated Values
DL	Deep Learning
FC	Fully-Connected
FFT	Fast Fourier Transform
GAN	Generative Adversarial Network
GAP	Global Average Pooling
GPS	Global Positioning System
GRU	Gated Recurrent Unit
HAR	Human Activity Recognition
IMU	Inertial Measurement Unit
LGO	Leave Group Out
LOCO	Leave One Condition Out
LOUO	Leave One User Out
LSTM	Long Short-Term Memory
LUO	Leave User Out
L&H	Laput and Harrison
MARG	Magnetic, Angular Rate, and Gravity
ML	Machine Learning
NOVA	Non-Verbal Annotator
OOD	Out of Distribution
OVA	One-vs-all
PPG	Photoplethysmogram
RNN	Recurrent Neural Network
ReLU	Rectified Linear Unit
SOTA	State-of-the-art
STFT	Short-time Fourier Transform

1 Introduction

With the advent of deep learning (DL), there has been an ever-increasing applications of deep learning-based models for the purposes of classification, forecasting, unsupervised learning. These applications have proliferated and penetrated the daily lives of people from automatic organisation of photo albums, criminal investigation, and physics simulations to stock market and weather predictions. These use cases have very real impacts on not just our personal lives but the organisation corporations and governments as well, emphasising the need for investigating the accuracy and reliability of such methods.

Machine learning (ML) has been the topic of research for several decades but it was only recently with the boost in computational resources and research in data-driven techniques that deep learning has shown tremendous promise. DL models have shown better performance than traditional machine learning approaches if provided with large enough and well curated datasets [1]. While traditional approaches rely on intricate feature engineering and work better on low dimensional data by performing classification in a higher dimension space, DL approaches can do the heavy lifting of learning the necessary features automatically from the higher dimensional training data and then performing classification in a lower dimension latent space. Since most of the data arduous for classification in the modern world consists of images, videos, and other higher dimensional data, DL has found popular use in consumer and industrial applications of image classification and such in recent years.

1.1 Robust Recognition on Smartwatches

A consumer-oriented domain gaining track recently is gesture and human activity recognition (HAR) on smartphones [2–6] and wearables [7–12], specifically smartwatches [13–16]. Consumer smartwatches come equipped with several capable sensors like inertial measurement units (IMU), global positioning system (GPS), photoplethysmogram (PPG) sensors enabling a lab-on-a-device type of environment for both data collection and monitoring a person's estimated position, posture, and activity. IMU or magnetic, angular rate, and gravity (MARG) sensor arrays are the bare minimum requirement for HAR in daily use and are even prevalent in the lower price bracket of smartwatches and fitness trackers.

The rising popularity of fitness trackers and smartwatches as an extension of the capabilities of smartphones, has found extensive use in exercise activity tracking like steps and repetitions counting, calorie expenditure estimates, and activity recognition. With so many humanly possible activities to keep track of, it becomes a matter of paramount importance that the recognition algorithm or model is able to robustly infer the user's intentions for a seamless experience while being able to reliably reject unintended gestures or false positives.

1.2 Deployability of Deep Learning-based models

The success of DL-based models in achieving high classification accuracies comes with a significant cost on the computational resources required to train, deploy, and continuously run these models for inference. A typical DL model is tasked with minimising a certain loss function to fit the ground-truth training data by backpropagating the gradients through its computational graph. This supervised learning setup usually uses human-labelled classes which the network is supposed to predict correctly after the training process. The trained model is then tested and deployed where the forward pass of the network is used for inference.

Several applications in a deployment scenario require a network's size to be on the conservative end with some trade-off on their robustness. The deployability is even more pressing an issue on low-end hardware devices which can only afford to spare a miniscule amount of compute operations and battery power. A model should be able to run continuously on this computationally impoverished setup with a minimal resource footprint.

1.3 Problem Statement and Contribution

The aim of this thesis is to explore the problem, "is it possible to have a robust and deployable deep gesture recogniser which is capable of running on a smartwatch itself". The definition of robustness can vary under different contexts so the thesis focuses on the problem of individual and situational variability as a quantification of robustness in different settings. It has been recently shown that HAR classification robustness is adversely impacted by a change as simple as re-wearing the smartwatch [16]. Therefore, it is important to define, quantify, and assess robustness for individual and situational variability in the context of human-centred applications.

A robust recogniser may be achievable with sufficient complexity in the model but is it capable of being deployed in a production environment, specifically on low-end devices like a smartwatch? Deployability of the model is then defined as the size or compute operations needed for inference on the smartwatch. It has been shown that even changing the sensor sampling rate can have significant effects on the battery life of the device [17, 18] and on the classification accuracy [16]. Therefore, there needs to be a balance between computational complexity — which is also dependent on the sensor data, and the power consumption on the wearable.

The thesis makes contributions for these two problem statements by addressing different model designs and training techniques for raw and preprocessed sensor data that favour deployability and a general sense of robustness for training a non-overfit and application-oriented classifier. The proposed classifier is presented to work for gesture, as well as "one-shot" activity recognition for the real-world scenarios of payment gesture recognition, and steering activity recognition. The thesis further contributes on model evaluation and interpretability using cross-validation methods to address individual and situational robustness of gesture and activity recognisers.

1.4 Research Questions

The thesis approaches this multi-faceted topic of robust gesture and activity recognition with the following research questions,

1. How does one define robustness in general?
2. What does robustness mean in the context of user experience?
3. How does one quantify robustness in this context?
4. What are the factors that contribute to robustness of a deep learning model?
5. How to train a computationally light deep learning model and measure its robustness in user-experience?

1.5 Thesis Structure

This thesis has the following structure: Section 2 delves into the literature and the state-of-the-art (SOTA) methods. Section 3 presents the methodological choices for the study. Section 5 discusses the results of the experiments defined in Section 4 and Section 6 finishes with the conclusion and future work.

2 Background

Recent literature in HAR has shown that the potential solutions to the problem of gesture and activity recognition can be approached from various angles. This section dives into the previous works and common techniques used to address HAR and advances in relevant domains.

Machine learning deals with the algorithmic approaches which adapt and improve with experience and new data to learn certain tasks. A model of the problem is used to make decisions using training data as its sample. Supervised machine learning uses data with ground truth labels and tries to fit the model to output the same label class for the input data during training. This process benefits substantially with well-crafted features extracted from the raw data rather than learning with the raw data itself. Otherwise the model might learn irrelevant relations or patterns in the dataset introduced due to statistical noise.

Deep learning takes machine learning one step further by automating the process of feature extraction thereby making it scalable. This also helps break the barriers of traditional machine learning approaches whose performance saturates due to scalability issues. Furthermore, advances in computational power during the last two decades has enabled deep learning to process huge amounts of data with proportionally large neural networks. DL-based models utilise and exploit patterns in input data distribution to generate better features at different levels of spatial complexity. This allows the model to relate the output to the high-dimensional input with minimal human intervention for crafting features.

2.1 Deep Learning for Activity Recognition

On-device recognition and deployment of deep networks has seen significant progress in the past decade with ongoing developments of optimised frameworks such as TensorFlow [19] and PyTorch [20] with recent focus on their mobile variants. These frameworks have enabled rapid prototyping of deep neural networks in research as well as in production environments. However, memory footprint, execution time, power consumption, and scalability remains a bottleneck on resource-scarce devices resulting in a trade-off between model performance and resource utilisation [21]. With this premise, the need for optimised networks to run on such low-end hardware is of utmost importance. Model compression helps reduce the storage space and memory footprint of otherwise large and practically infeasible deployment of networks for such devices [22]. Techniques like quantisation to train networks with low floating point precision weights and activations have also shown improved inference and computation times with virtually no loss in classification accuracies on unoptimised GPU hardware [23–25].

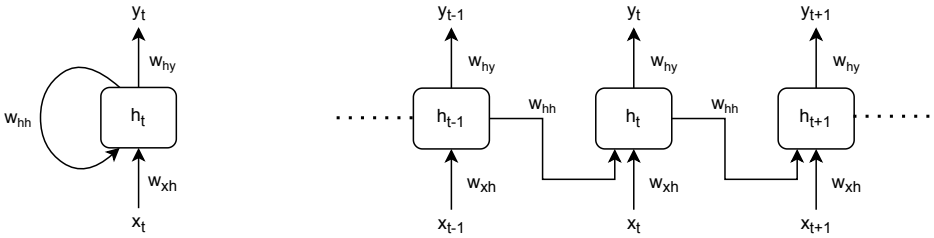
HAR primarily utilises data from various sensors like IMU, MARG, PPG etc. for forecasting or recognising user activity. This data is usually treated as a multivariate time-series streamed from the location of interest to a processing unit which processes it and uses it for inference. Due to the nature of the dimensionality of the data, a multitude of deep learning approaches exist for processing and inference.

2.1.1 Recurrent Neural Networks

Recurrent neural networks (RNN) find common use in time-series data ranging from stock-market prediction to natural language processing and machine translation. A vanilla RNN consists of a hidden state h_t and takes an input sample x_t at time t and yields an output y_t and an updated state h_{t+1} as shown in Equations 1 and 2. The hidden state h_t is used with the input at the next time-step x_{t+1} and the process repeats sequentially. This procedure is commonly described as the "unrolling" of the RNN as depicted in Figure 1.

$$\mathbf{h_{t+1}} = \sigma_h(W_{xh}\mathbf{x_t} + W_{hh}\mathbf{h_t}) \tag{1}$$

$$\mathbf{y_t} = \sigma_o(W_{hy}\mathbf{h_{t+1}}) \tag{2}$$



(a) RNN loop. (b) Unrolled RNN.

Figure 1: Representation of the RNN operations and their "unrolling".

The sequential nature of RNNs has the advantage of utilising temporal context making them ideal for use in sequences of arbitrary lengths. Vanilla RNNs tend to be even less popular because of their problems of vanishing gradients over long sequences of data [26]. More advanced variants like Long Short-Term Memory cells (LSTM) [27] and Gated-Recurrent Units (GRU) [28] skirt around this issue with the additional overhead of having to maintain extra internal states. Equations 3 to 8 present the formulation for an LSTM cell.

$$\mathbf{f_t} = \sigma(W_f\mathbf{x_t} + U_f\mathbf{h_{t-1}} + b_f) \tag{3}$$

$$\mathbf{i_t} = \sigma(W_i\mathbf{x_t} + U_i\mathbf{h_{t-1}} + b_i) \tag{4}$$

$$\mathbf{o_t} = \sigma(W_o\mathbf{x_t} + U_o\mathbf{h_{t-1}} + b_o) \tag{5}$$

$$\tilde{\mathbf{c}}_\mathbf{t} = \tanh(W_c\mathbf{x_t} + U_c\mathbf{h_{t-1}} + b_c) \tag{6}$$

$$\mathbf{c_t} = \mathbf{f_t} \cdot \mathbf{c_{t-1}} + \mathbf{i_t} \cdot \tilde{\mathbf{c}}_\mathbf{t} \tag{7}$$

$$\mathbf{h_t} = \mathbf{o_t} \cdot \tanh(\mathbf{c_t}) \tag{8}$$

where x_t is the input at time t, hidden state is h_t, forget gate f_t, input gate i_t, output gate o_t, c_t cell state, and W, U, and b are the weights and biases. These operations are summarised in Figure 2.

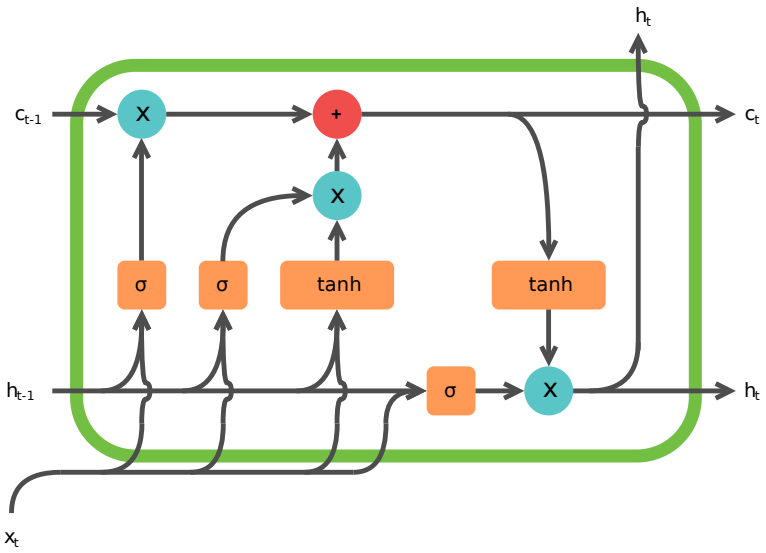


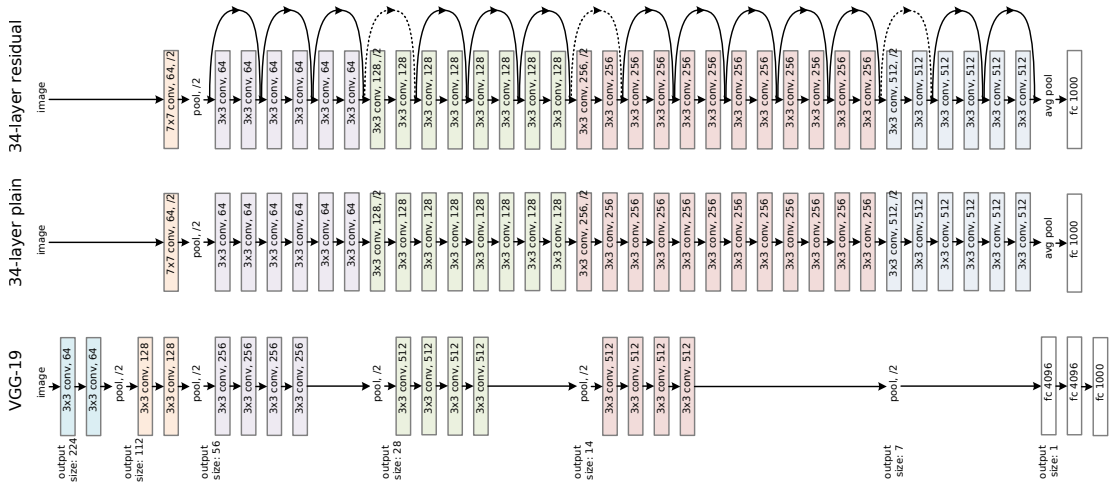
Figure 2: Operation flow of an LSTM cell at a single time-step.

However the operations of RNNs on an arbitrary time-series segment tend to be less parallelisable because of their sequential nature which makes them inherently slower for both training and inference. However, with an off-device recogniser and enough resources for streaming the sensor data to a processing unit, this can be mitigated to some extent. Gesture and activity recognisers in this direction tend to primarily use LSTMs as base-learners [29, 30] because of their advantages over vanilla RNNs or some Convolution + LSTM (ConvLSTM) variants utilising intermediate features [11, 31] but their strength lies in processing temporal signals of arbitrary lengths.

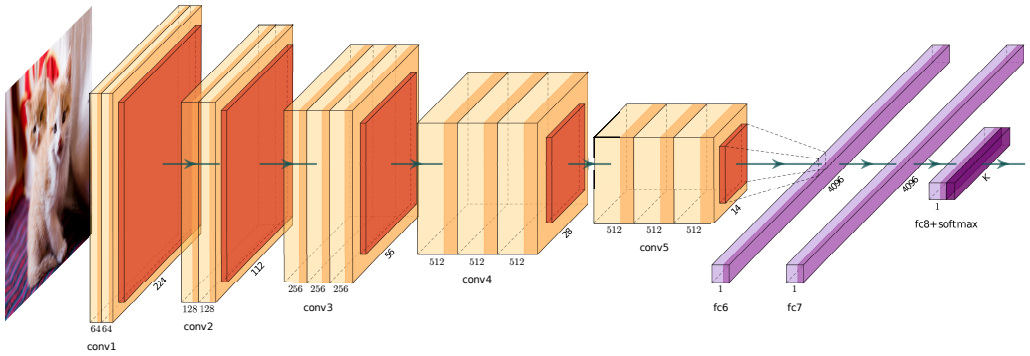
2.1.2 Convolutional Neural Networks

Convolution neural networks (CNN) use learnable filters or kernels with shared weights reducing their size significantly in comparison to fully-connected (FC) dense networks. They work as excellent feature extractors and are used widely in image classification tasks with the most notable architectures being VGG16 [32] and ResNet50 [33] as shown in Figure 3.

Taking an example of CNN for image classification, a kernel is slid across the input image with element-wise multiplication generating an output tensor, called the output feature, which is then used for further processing or for classification depending on the depth of the network. An example of the 2D convolution operation, widely used in image data, is shown in Figure 4 and can be represented by Equation 9. The size of the output feature map is dependent on the size of the kernel used, the padding applied to the input, as well as the stride of the convolution operation. This can be calculated by Equation 10 which shows the size of the output along the horizontal axis (assuming symmetric stride s and padding p).



(a) Architectures for the VGG and the ResNet [33].



(b) VGG-16 architecture for image classification.

Figure 3: Architectures of state-of-the-art image classification deep neural networks.

$$O_{m,n} = \text{conv}(I, K)_{m,n} = \sum_i \sum_j \sum_k K_{i,j,k}\, I_{m+i-1,n+j-1,k} \tag{9}$$

$$\dim(O_{m,n})_x = \left[\frac{\dim(I)_x + 2p - \dim(K)_x}{s} + 1 \right] \tag{10}$$

The output of the convolution operation typically undergoes an activation function (ReLU, TanH) and a pooling operation like max-pool or average-pool which take the maximum value or the average of all values in their scope respectively. These pooling operations are depicted in Figure 5.

The output of the CNN layers is of sufficiently reduced dimensionality and is capable of being used with FC layers. This leads to the commonly used notion of terms like “feature extractor” or “head” for the CNN layers and “classification layers” for the later FC with CNNs finding rigorous use in transfer learning [34]. A large network like the VGG16 is trained on a large publicly available dataset like the ImageNet [35] and the pretrained weights of this model are reused by keeping the initial convolution layers frozen, allowing them to act as feature extractors. The

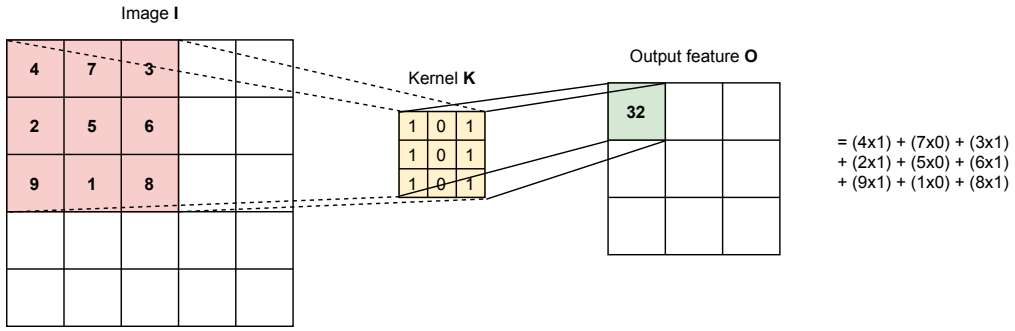
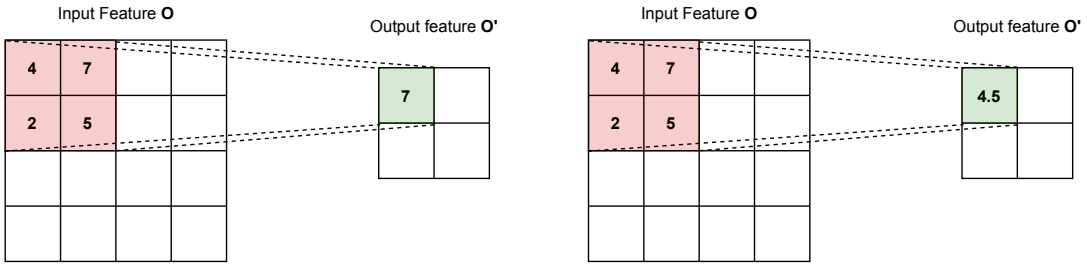


Figure 4: Unpadded 2D convolution operation for image classification tasks. This process is repeated by striding the kernel across the image.



(a) Max-pool (b) Average-pool

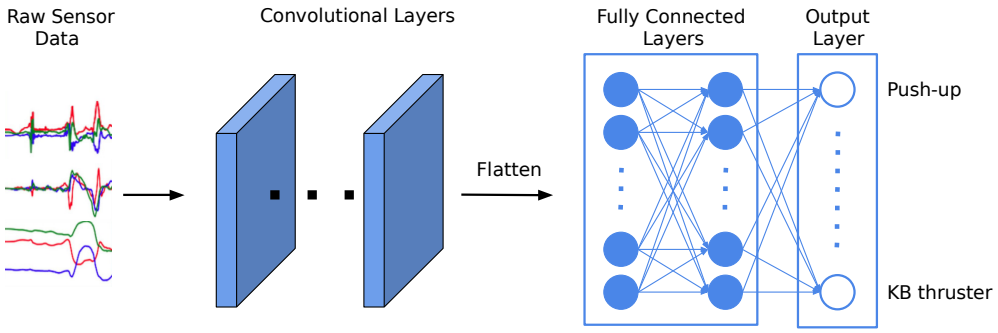
Figure 5: Common 2D pooling operations (scale factor 2, stride 2) applied on output features obtained after 2D convolutions.

later FC layers are then trained to fine-tune or adapt to another classification task for some similar dataset.

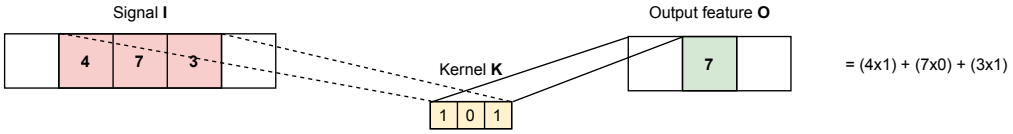
Just like with 2D images, CNNs can be used with multichannel 1D temporal signals as feature extractors. The stride and kernels in this case would be just along one direction as shown in Figure 6. The main caveat of CNNs is that they work only for fixed-sized inputs. This is a potential issue for temporal input as it can of arbitrary length. This has been addressed by taking fixed-sized windows of a predetermined duration and then using the latent representations with RNNs [7, 11, 29], with attention [7, 36], or with temporal convolutions [37]. Approaches using direct CNN outputs from windows for exercise activities have also shown remarkable classification performance [15] as depicted in Figure 6a. Spectrogram has also been used to interpret the 1D signals as 2D images with temporal and frequency components for audio and sensor classification [16, 38–40].

2.1.3 Sensor and Modality Fusion

A device uses multiple sensors that are used in conjunction to provide relevant measurements. For example, an IMU contains a gyroscope and accelerometer which output angular velocity and linear acceleration respectively. Just from these measurements from the IMU it is possible to provide estimates of relative direction, device orientation, and crude position. This process of using multiple sensors to derive



(a) Convolutions used for processing sensor signal windows [15].



(b) Unpadded 1D convolution operation for sequential data. This process is repeated by striding the kernel horizontally across the sequence.

Figure 6: Approaches to directly process the time-series sensor data using CNNs.

meaningful estimates of measurements of the device's state is called sensor fusion [41]. These measurements can be complimented by adding new sensors or with the addition of the same sensor to reduce uncertainty. IMUs are commonly found in consumer devices like smartphones, smartwatches, fitness trackers, and virtual reality headsets and are used primarily for estimating orientation. Several sensor fusion algorithms exist to enable attitude estimation (AHRS) by combining accelerometer and gyroscope or additionally with a magnetometer like the complementary filter, extended Kalman filter, Madgwick filter [42], and deep learning-based approaches like RIDI [43] and RoNIN [44]. Developments in this direction has boosted the applications of HAR on both smartphones [4–6, 45–48] and smartwatches [49, 50], and also in novel applications like sign language recognition [51] and gyropens [52].

One of the simplest sensor fusion methods, the complementary filter is shown in Figure 7. With the help of an accelerometer, one can deduce the angle the device makes with respect to the ground and using a gyroscope, the angular velocity of the device can be obtained. However, accelerometer readings have high frequency noise components while gyroscope measurements accumulate drift on numerical integration. This is why a low pass and a high pass filter are used on these processed sensor signals respectively to obtain angle estimates from both. A simple weighted sum then provides us with a more refined and stable angle estimate. Both of these individual sensors operate in the device coordinate system, that is, their zero reading is set from the moment the device is turned on. So for improving the estimate further, a magnetometer is added which helps the accelerometer by providing the heading direction in the global coordinate system.

Modality fusion on the other hand uses information from multiple different

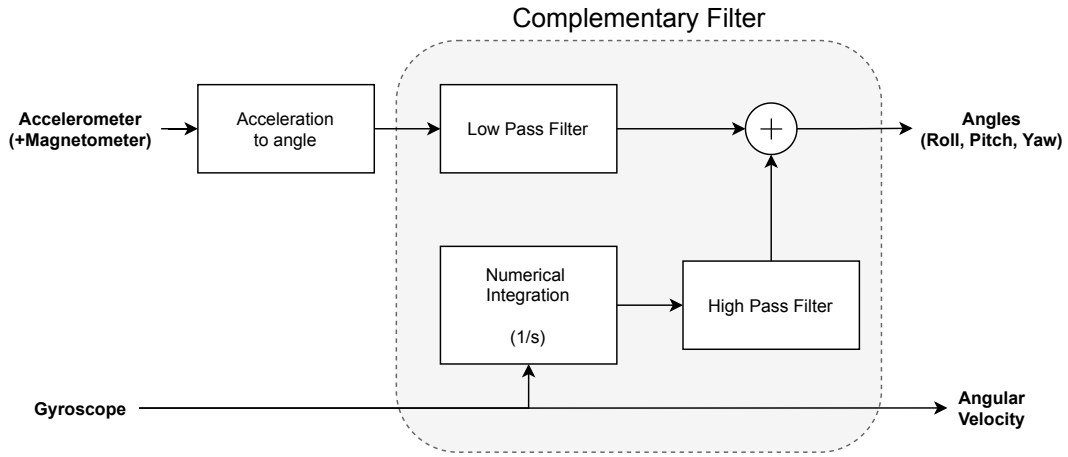
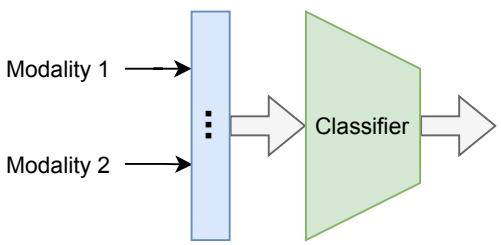
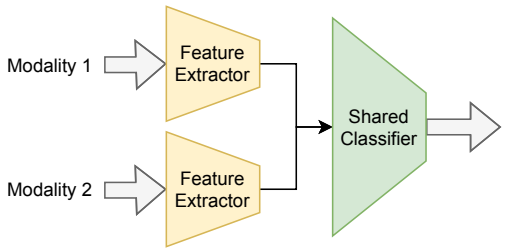


Figure 7: Complementary filter sensor fusion using an IMU/MARG sensor setup.

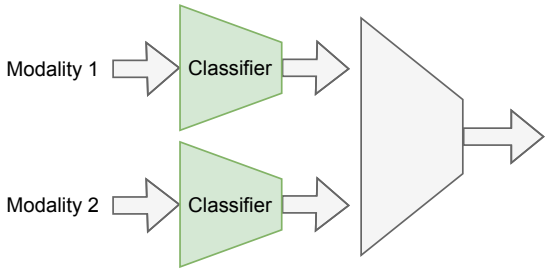
unimodal representations [53]. Audio-video data and synchronised text is one of the most commonly applied topics in this domain. These modalities are fused at different points in the deep network and are therefore classified into early, late, and mid-level fusion policies [54, 55] as shown in Figure 8 or using advanced cross-modal preserving methods like tensor fusion [56]. This makes it possible to process inputs from different sensor sources or with different transformations and preprocessing applied pre-classification [12].



(a) Early fusion.



(b) Mid-level fusion with shared classifier.



(c) Late fusion using the outputs of unimodal classifiers.

Figure 8: Different fusion policies for inputs to a multimodal network.

2.2 Spectrograms

Spectrograms are extremely common in the domain of audio processing for classification and chirp detection [57–61]. Owing to the similarity of audio data and motion sensor data, since both are temporal, sequential, and voltage values, spectrograms have also been extremely popular for in HAR and gesture recognition [12, 13, 16, 30, 39, 62, 63]. A spectrogram allows the time-series data to be interpreted as an image which can than be leveraged by using state-of-the-art image classification network architectures like the VGG16 as shown in a study by Laput and Harrison to achieve exceptional HAR accuracy [16]. Figure 9 shows Laput and Harrison's VGG16 inspired model for HAR.

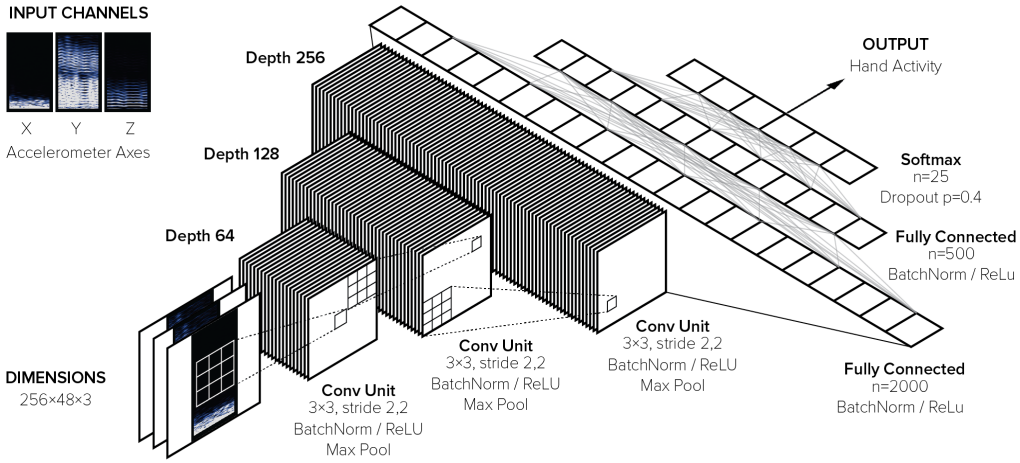


Figure 9: VGG16-based HAR model proposed by Laput and Harrison.

2.2.1 Spectrogram Generation

A spectrogram is generated by taking a Short-time Fourier transform (STFT) over a sliding window across a time-series signal. The different parameters chosen for this process, along with the length and sampling frequency of the signal determine the output size of the spectrogram. A discrete time STFT for a discrete signal $x[n]$ is defined by multiplying it with a window function and taking the Fast Fourier Transform (FFT) of the resulting signal. This window slides along time to give the STFT of the signal.

$$X(k,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-k]e^{-j\omega n} \tag{11}$$

k represents the rolling frame along the temporal dimension and ω the frequency bin from the FFT. The spectrogram then is the power spectral density of this complex-valued function by taking its squared magnitude.

$$X_{PSD}(k,\omega) = |X(k,\omega)|^2 \tag{12}$$

The sliding window function used in STFT is preferred to have a non-negative value in the centre which gradually tapers to zero towards the window boundaries. If non-zero values are present at the boundaries, it leads to harsher artifacts due to the overlap process. For this reason, a smooth and periodic window function is preferred [64]. However, this results in a trade-off in the spectral localisation because the multiplied signal on which the FFT happens appears smoother than the original signal.

One commonly used window that satisfies these properties is the Hanning window function which is a weighted cosine dropping smoothly to zero as it approaches the boundaries. For a window of size n_{win} and $0 \leq n \leq M$

$$w[n] = \frac{1}{2}\left[1 - \cos\left(\frac{2\pi n}{n_{win}}\right)\right] \tag{13}$$

For simplicity, the number of samples to consider for the STFT while generating the spectrograms is set equal to the window size for minimising artifacts.

$$n_{fft} = n_{win} \tag{14}$$

The FFT algorithm really benefits in speed if the window sizes are in powers of 2 with a Big-O complexity of $O(n \log n)$ [65, 66]. The number of frequency bins obtained from STFT is then given by

$$n_{bins} = \frac{n_{fft}}{2} + 1 \tag{15}$$

In an unpadded signal, the centre of the window function corresponds to the ${}^{n_{win}}/_{2}{}^{\text{th}}$ index in the original signal. This may or may not be desired since there is a loss of information around the boundaries in the resultant signal. It is more convenient if the centre value of the window function corresponds to the 1st value of the original signal. Similarly, the last section of the original signal does not receive the same number of overlaps if just beginning of the signal is padded. So, a symmetric padding on both sides of ${}^{n_{win}}/_{2}$. However, this leads to artifacts around the edges of the spectrogram [64].

The overlap of the time-series data between two consecutive window shifts, usually described in the form of overlap percentage or hop size, is a critical parameter. The overlap dictates the trade-off of certainty in time versus frequency. A huge overlap approaching 100% is analogous to just taking the FFT of the entire signal and results not only in data redundancy, but also in aliasing. For this reason, an overlap of 50% is recommended for most periodic windows with an upper limit of 75% [67] for robust reconstruction and preventing aliasing.

The overlap and hop size are described by the relation

$$n_{hop} = \left[1 - \frac{\text{overlap}\%}{100}\right] n_{fft} \tag{16}$$

Figure 10 shows spectrograms generated with different mentioned parameters.

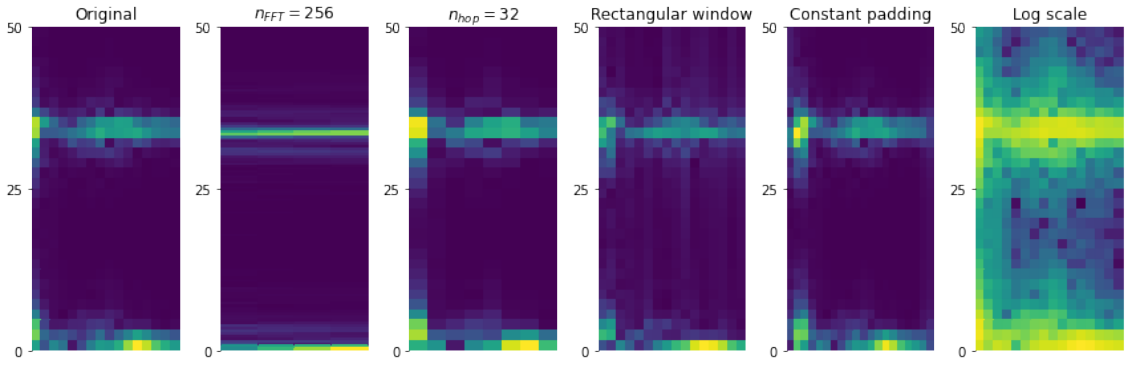


Figure 10: Spectrograms generated with different parameters. The original spectrogram is generated with $n_{fft} = 64$, $n_{hop} = 16$, Hanning window, no padding, and on linear scale. Each spectrogram shows the effect of changing a single parameter while keeping the rest same as that of the original.

2.3 Robustness

In applications of HAR using sensor data, it may be more appropriate to evaluate a technique's robustness to changing input distributions, noise, and out-of-distribution (OOD) inputs. Measuring the reliability of such classifiers becomes paramount in these settings which have frequent changes in situations, tasks, or operators. DL models in particular have been shown to be less robust to input noise, OOD samples, and poor features [68–70]. Further, it has been shown that achieving robustness is at odds with our desire for a recognizer that has high accuracy [71–73]. So, we have to ensure a balance between the recogniser performance and its robustness to OOD inputs. Significant efforts have been made to address this problem, general to DL-based classifiers, by modifying the training methodology [74], encouraging classification boundary smoothness [72, 75, 76], or by using loss functions which encourage robustness [77, 78].

The definition of robustness varies with the context and the parameter for which the variability of the model is trying to be minimised. In a real world HAR scenario, a typical user is concerned with their perception of the reliability in response to their input under different situations, i.e., how reliably can the application produce the same output for similar input under different conditions. Consequently, for a user-oriented application, it is reasonable to optimise a gesture recogniser for robustness in recognition accuracy across different demographics and changing situations.

2.3.1 Overfitting and Regularisation

Robustness in DL is primarily concerned with resilience of the network's predictions to outliers. These outliers may be from unseen test data with slight distributional shifts — which is usually the case since test distributions tend to differ from training distributions [79] or maliciously perturbed samples with the aim of "fooling" the network to misclassify a negative sample as a positive one especially in user authentication. The latter acts are commonly known as adversarial attacks and

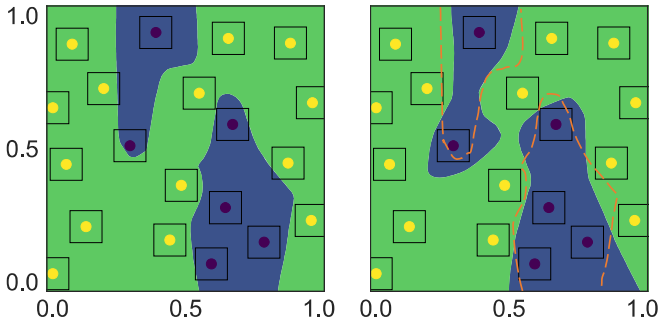
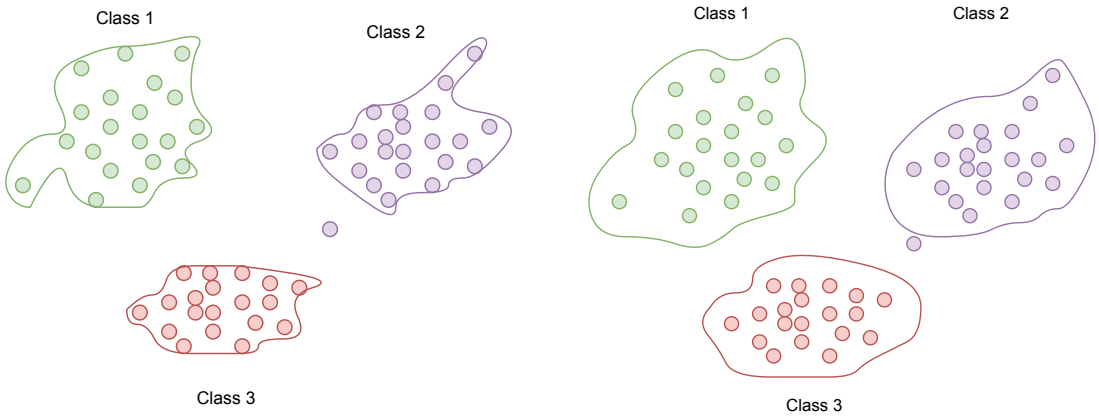


Figure 11: Motivation of TRADES algorithm for promoting boundary smoothing [72]. Left, decision boundaries of a normally trained classifier. Right, smoother decision boundaries of a classifier trained with taking boundary smoothness into consideration via TRADES loss function.

the robustness associated with such malicious samples is thereby called adversarial robustness. A multitude of reasons have been demonstrated to be at play for a network's failure for adversarial perturbations like high bias with negative local effects due to regularisation [80], or more empirically, overfitting [81] which seem to be contradictory. Yet another study finds that an overfit network is sufficiently vulnerable for privacy-breach attacks which is shown to be analytically controlling for overfitting by regularisation and design of the model architecture itself [82].

Overfitting is a concept in statistical models where the underlying decision function fits too closely to a specific set of data points, generally to the training set, severely impairing the model's ability to generalise to unseen test data which most likely will have a different distribution. A well-fit model on the other hand helps the model to have relatively lax decision boundaries such that the model is robust to slight distributional shifts for the test data. This can be achieved by modifying the training process like early stopping [81], model design choices [82] or even by accommodating a penalty for sharp decision boundaries during training [72]. Figure 12 shows a representation for the difference between an overfit and a well-fit multi-class classifier.

Regularisation techniques like dropout in FC have shown significant benefits to combat overfitting [83]. Due to the enormous size of the FCs, they are severely prone to overfitting. A viable alternative to using FCs for the flattened feature vector after CNN feature extraction is by using global pooling, specifically Global Average Pooling (GAP) which forces the CNNs to be trained in such a way so that they output feature scores for classification rather than high dimensional features [84]. Equation 17 and 18 represent the GAP operation for one-dimensional and two-dimensional convolution features respectively.



(a) An overfit classifier.

(b) A well-fit classifier.

Figure 12: Representation of decision boundaries on the training set for multi-class classifiers under different fitting conditions for the same set of data points.

$$\bar{\mathbf{x}}_{n,i} = \frac{1}{J}\sum_{j=1}^{J}\mathbf{x}_{n,i,j} \tag{17}$$

$$\bar{\mathbf{x}}_{n,i} = \frac{1}{J}\sum_{j=1}^{J}\frac{1}{K}\sum_{k=1}^{K}\mathbf{x}_{n,i,j,k} \tag{18}$$

Further, Training of a deep neural network is typically sensitive to hyperparameter initialisation. Normalisation methods like batch-norm [85], instance-norm, layer-norm [86], and group-norm [87] have been shown to provide sufficient robustness to the choice of hyperparameters. Figure 13 shows the difference between these normalisation strategies. Although it is still a topic of debate whether the benefits are due to the improvement in internal covariate shift as initially argued or due to factors like optimisation landscape smoothing [88, 89].

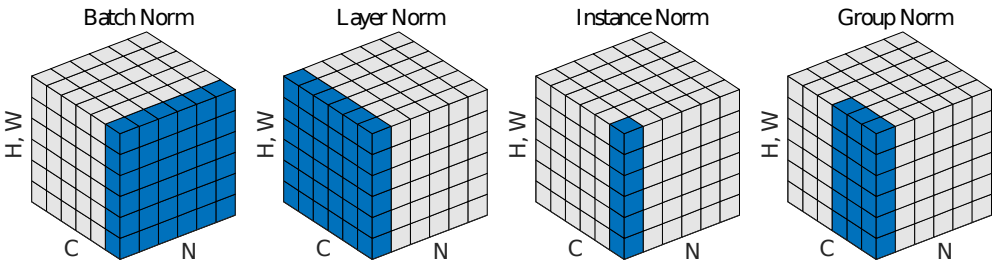


Figure 13: Different normalisation strategies for a batch of inputs. N is the number of samples in the batch, C is the number of channels in a sample and H, W represent the spatial axes.

2.3.2 Data Augmentation

Data augmentations are sets of techniques for modifying or slightly perturbing the ground truth samples in order to improve a model's ability to generalise better and to be more robust to slight variations in the input. This is especially helpful when the training dataset is relatively small or their is a class imbalance which could lead to bias in the model or poor inference performance at deployment due to overfitting. Augmentation thus helps to artificially inflate the dataset to mitigate overfitting and is one of the most popular remedies applied as a method of regularisation [90].

Data augmentation techniques for photographic images are usually in the form of random cropping, rotating, perspective transforms, filtering, normalisation [91], adding noise etc. which are nowadays baked into deep learning frameworks for ease of use. More advanced methods like random erasing [92] and style transfer or synthetic data generation with Generative Adversarial Networks (GAN) [93, 94] have successfully demonstrated the advantage in boosting the classifier performance for their respective tasks.

However, images are inherently different from STFT spectrograms. Spectrograms have a sequential nature as one traverses along the temporal axis because of the overlap used during the generation process. Further, the values of the spectrogram are usually in a log or decibel scale to emphasise a human's perception of auditory frequencies or musical notes. Therefore it makes it difficult to use the same augmentations that are commonly used for spatial image data. Naively applying methods like time-shift, frequency-shift, and image warping have showed detrimental results on audio spectrograms by Nanni et al [57]. The same study demonstrates marginally better performance if appropriate augmentations are applied to the raw audio signal first and the spectrogram generated afterwards. Further, patch erasing with frequency and time masking has been shown to improve speech recognition using spectrograms [95]. Figure 14 shows some of the discussed spectrogram augmentation techniques.

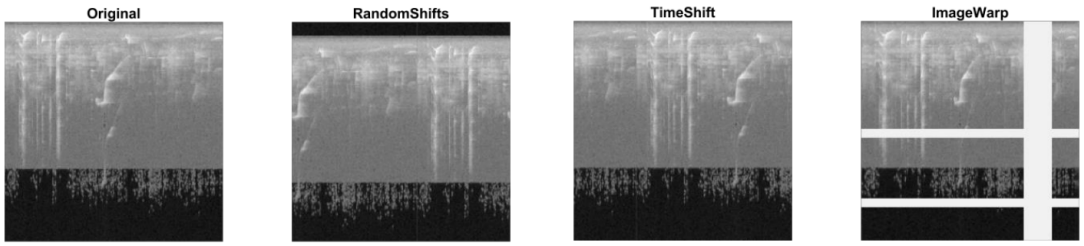


Figure 14: Augmentation strategies applied on the spectrogram data rather than on the original sound signal [57].

Unlike images and audio, sensor data augmentation has had less exhaustive research but because of the similarity to audio data, some augmentations may be directly applicable on sensor data as well like those by Nanni et al. In an HAR study, Eyobu et al [30] propose downsampling using local averaging and feature shuffling as a method for augmenting IMU sensor data. SensoryGAN has also been proposed as an advanced method to generate and augment sensor data for HAR applications [96].

2.3.3 Cross-validation for Model Evaluation

Classical datasets in ML and DL for classification follow a categorical setup with a set of data points belonging to a certain class A, then another set belonging to class B and so on. The division is based on classes, hence, the problem of classification. The de-facto way to train a model with such a dataset involves splitting the dataset into three partitions — training, validation, and testing. The model is adjusted with the training dataset and its performance monitored with the validation dataset during training. The validation split provides a good estimate of when the model starts to overfit. The test dataset is then used post-training for a final evaluation of the trained model on unseen data. This is called the three-way holdout method for training.

However, it may be more appropriate to partition the dataset into smaller partitions and cycle those splits as training, validation, and testing one by one to get a better overview. k-fold cross-validation achieves this by splitting the complete dataset into k partitions. $k-1$ partitions are used for training and the remaining partition is used for validation [97]. This is repeated k times (folds) with different combinations of the splits as shown in Figure 15. Performance on the k validation sets is then used to infer the average model behaviour. This ensures that each sample in the dataset is used both for training and validation. Despite its advantages, k-fold cross-validation remains less popular than the hold-out due to its high computational demands.

Real world data, especially that generated from humans, can have more than one level of stratification. This hierarchy also means less data per individual class at the lowest level. This puts into question a model's performance, ability to generalise, and robustness under new situations, individuals, and other external factors. The simple three-way holdout does not provide nearly enough information on model capabilities and may seemingly suppress or downplay certain situations where the model fails considerably. k-fold cross-validation can be modified to address these concerns by instead treating each user or situation as a partition for the method as shown in Figure 15. Then k-fold can be applied as leave-user-out (LUO) or leave-group-out (LGO) to aptly reflect the stability of the model under varying strata. The performance statistics would then reflect the model's robustness and its potential capabilities on an unseen group when trained with the complete dataset. A study by Laput and Harrison [16] has shown this as an important measure for a model's performance under such an experimental setup where they portrayed the degradation of their model's performance under changing situations — or their model's robustness to situational variations.

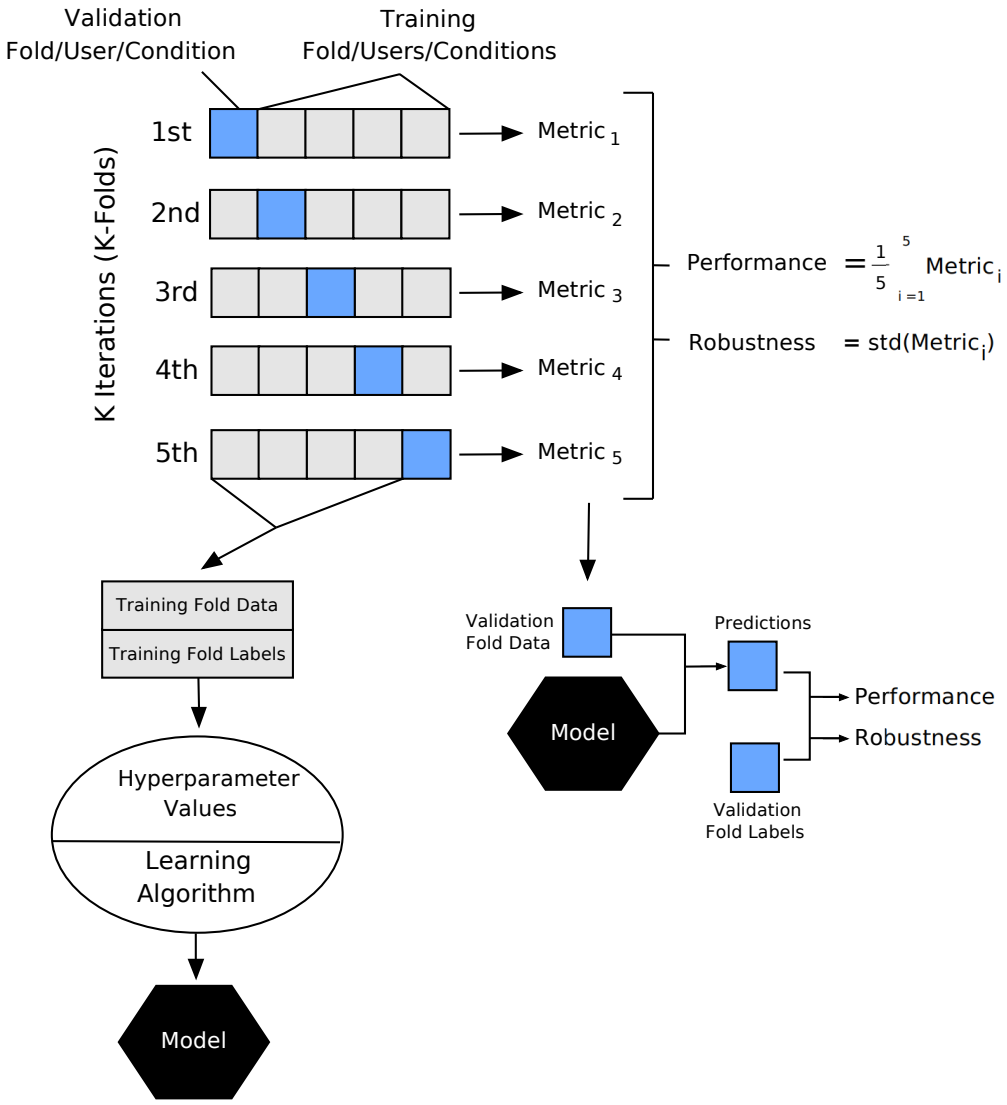


Figure 15: k-fold cross-validation and adapting it for stratified datasets [97].

3 Research Material and Methods

The aim of this thesis is to deploy a robust recogniser on a smartwatch with a model architecture that is capable of performing inference on the raw sensor data with minimum overhead. Considering the meagre computational resources available on a smartwatch, the preprocessing steps, the preprocessing steps as well as the operations in the classifier itself have to be minimised as much as possible. Acknowledging the state-of-the-art results achieved by using high resolution spectrograms by Laput and Harrison (L&H) [16], a similar architecture is devised but with a smaller footprint. These architectures can then be benchmarked against the classifier that works on the raw sensor data and the L&H baseline.

A simpler model, with a smaller number of trainable parameters, is less prone to overfitting. This translates to smoother, less tight decision boundaries. Further, architectural decisions on the incorporation of certain layers have a significant impact on controlling the number of model parameters. With these motivations in mind, an appropriate pipeline and test-bench are prepared for the training of situational- and individual-invariant robust gesture and activity recognisers.

3.1 The Dataset

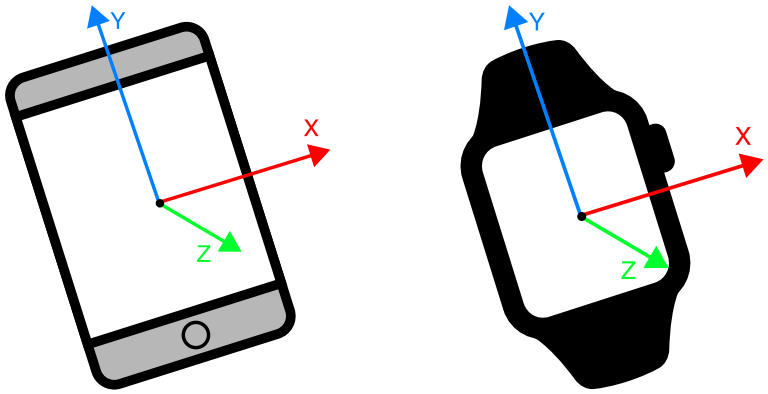
The custom dataset used for the experiments consists of data from 24 participants for several gestures and activities. The data is recorded using a Ticwatch Pro 3 smartwatch sampled at approximately 100 Hz and contains sensor data from accelerometer, gyroscope, and gravity sensor from the Android Sensor API in the Comma-Separated Values (CSV) format. The annotations from Non-Verbal Annotator (NOVA) [98, 99] are available as time stamp durations when the gestures or activities occur for each class and for all levels of stratifications. The sensor data has readings in SI unit and their coordinate system is shown in Figure 16. Extracts from the CSV file and the annotation files are shown in Listings 1 and 2 respectively. The start of the gesture or activity is considered to be between the process of the participant raising their hand and the end treated as the time between the process of lowering their hand as shown in Figure 17. The overall dataset structure is shown in Figure 18 and a description of the files in Table 1.

```
0.84544194;6.9000516;7.1132083
0.8334668;6.557564;7.302415
0.81909674;6.550379;7.3886356
```

Listing 1: Extract from semicolon delimited CSV of accelerometer sensor data. The three values in each row correspond to the X, Y, and Z -axis values for the sensor. Each row occurs after 10 milliseconds (100 Hz sampling rate).

```
1.28;2.72;0;1
2.72;20.48;3;1
20.76;26.44;1;1
```

Listing 2: Annotation file from NOVA. Each row corresponds to a gesture or activity instance with start timestamp, end timestamp, class index (internally used by NOVA), and confidence level (1 for manually labelled data).



(a) Android smartphone. (b) WearOS smartwatch.

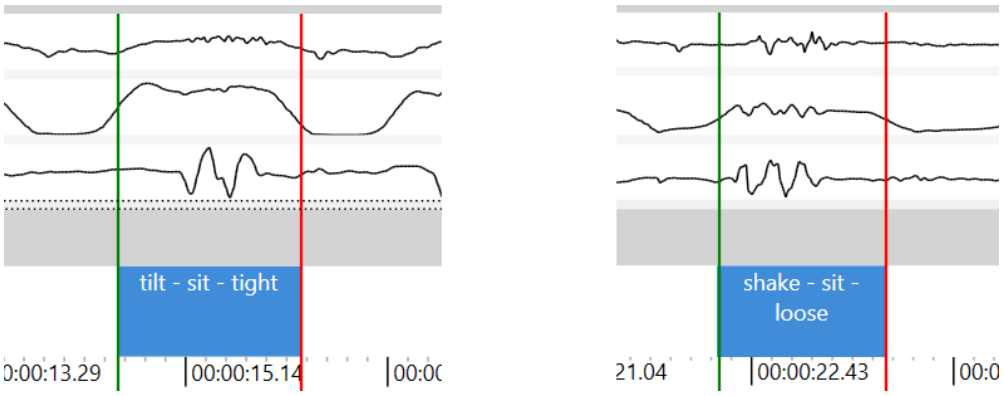
Figure 16: Sensor coordinate systems for the Android Sensors API.

The file naming convention for the annotation files follows the format,

$$\underbrace{\text{kh008}}_{\text{user}}\_\underbrace{\text{tight03}}_{\text{instance}}\_\underbrace{\text{payment tilt}}_{\text{experiment}}.\text{annotation}(\sim)$$

and the corresponding sensor files are named with the convention,

$$\underbrace{\text{trim}}_{\text{prefix}}\_\underbrace{\text{kh008}}_{\text{user}}\_\underbrace{\text{tight03}}_{\text{instance}}\_\underbrace{\text{Accelerometer}}_{\text{sensor}}.\text{csv}$$



(a) Tight-strap tilt gesture showing higher activity in gyroscope.

(b) Loose-strap shake gesture showing higher activity in accelerometer.

Figure 17: Annotation samples showing the starting and ending point of gesture. From top to bottom — accelerometer X-axis, gravity sensor Z-axis, gyroscope X-axis, annotation name with levels.

3.1.1 Payment Gesture

The collected gesture data is for two payment gestures — shake and tilt as shown in Figure 19. For each condition, both the gestures are performed 10 times per

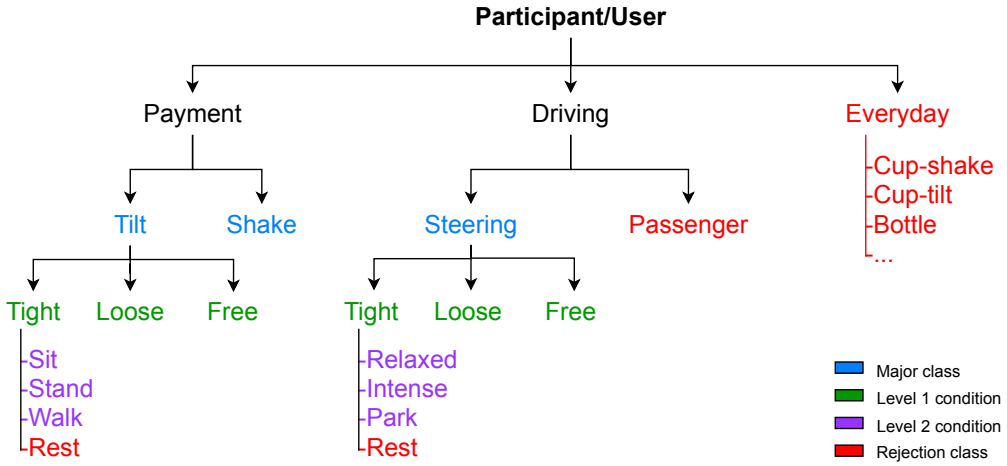


Figure 18: Structure of the dataset for a particular user.

Table 1: Description of the files in the dataset.

Extension format	**Source**	**Description**
`.csv` (Comma-separated Values)	Data collection tool	Sensor data in CSV files for the corresponding sensor (accelerometer, gravity, gyroscope). The data is actually semicolon-separated and not comma-separated for compatibility with NOVA.
`.annotation` (XML)	NOVA	Describes the annotation labelling scheme.
`.annotation` (tabular, text)	NOVA	Labels with start and end timestamps (in seconds), label ID from the annotation labelling scheme, and confidence value (not used).
`.nova`	NOVA	NOVA project file for editing and viewing if required.
`sampleRate.txt` (tabular, text)	Annotator	Contains the sample rates used for aligning the different sensor data in NOVA for annotation.
`.ftr` (Feather)	Processing	Contain the data frame for the user. "df.ftr" contains the intermediate multilabel representation, "mlb_df.ftr" contains the final multi-hot representation.
`.dvc`	DVC (Data Version Control)	Tracks the changes to the files in the dataset for a git-enabled version control.

participant resulting in a total of 90 instances of each gesture per participant. The gesture dataset is stratified into two different levels as shown in Table 2.

The "*free*" condition denotes an instruction for the participant to act out the gesture as they would naturally and with some intended variability while wearing the watch at the tight strap level. It sits on the same high priority level as watch tightness because L&H have previously shown post-watch-removal situations to be an important factor affecting model robustness. These post-watch-removal situations could have been at either ends of the tightness spectrum but they more importantly signify the effect of variability which the "*free*" class represents well.

(a) Shake gesture.

(b) Tilt gesture.

Figure 19: The gesture motions used as triggers for the intent of performing payment via the smartwatch.

Table 2: Description of the gesture dataset.

Level	**Parameter**	**Condition**	**Description**
1	Watch tightness	Loose	Watch worn with loose strap
		Tight	Watch worn with tight strap
		"*Free*"	Gesture performed naturally
2	Body posture	Sitting	Gesture performed while sitting in a chair
		Standing	Performed while standing upright
		Walking	Performed while walking around in the room

3.1.2 Steering Activity

The activity dataset is concerned with recognising the activity of steering while driving a car. The data has the same sampling rate of approximately 100 Hz and is recorded with participants driving in an in-lab environment using a gaming steering wheel while driving on a simulator. The setup is shown in Figure 20. The activity dataset is stratified similar to the gesture dataset as shown in Table 3.

(a) Steering simulator setup.

(b) Participant using the simulator.

Figure 20: Setup for recording the steering activity with a simulator.

Table 3: Description of the gesture dataset.

Level	Parameter	Condition	Description
1	Watch tightness	Loose	Watch worn with loose strap
		Tight	Watch worn with tight strap
		"*Free*"	Gesture performed naturally
2	Driving situation	Relaxed	Casual driving across the simulator map
		Intense	Driving to reach destination in a hurry
		Parking	Parking the car on the street

3.1.3 Rejection class

The dataset further comprises an overall rejection superclass collected in the lab which contains everyday gestures and activities meant to differentiate intended user actions from unintended ones. The "everyday" class has no levels of stratification and is described in Table 4. An additional subclass called "rest" is labelled for durations where the participant does not perform any significant gesture or activity. However, these durations of inactivity are treated differently from the "everyday" superclass.

Table 4: Sixteen gesture and activity classes recorded under the "everyday" superclass.

Gesture/activity class	Description
cup - shake	Shaking a cup to confuse with the payment shake gesture.
cup - tilt	Tilting a cup to confuse with the payment tilt gesture.
wave	Waving hand to greet someone.
jog	Pretending to jog in place.
comb	Combing or pretending to comb their hair.
phone	Using their mobile phone.
bottle	Opening or closing the cap of a bottle.
drink	Pretending to drink from a cup.
knife	Cutting food with a knife and a fork.
fork	Pretending to eat with a fork.
spoon	Pretending to eat with a spoon.
burger	Pretending to eat a burger.
dust	Cleaning the table with a paper towel.
dishes	Pretending to wash a plate.
washer	Putting utensils into an imaginary dishwasher.
door	Opening and closing a door.

3.2 Data preparation

The raw sensor data was provided as three CSV files for each sensor along with separate files for the labelling scheme and annotations (units in seconds). For preparing the dataset for use in machine learning, the label files were parsed to obtain

the row indices of the samples in the CSVs by the simple relation in Equation 19 where i denotes the row index (rounded to the nearest integer) in the CSV file, t is the timestamp in the annotation file, and f_s is the sampling rate for the experiment.

$$i = \lfloor t \times f_s \rceil \tag{19}$$

The files are read with Python and an intermediate Pandas dataframe is created for each user (participant) to help load their data easily. The structure of the dataframe is shown in Table 5 where *user* is the code for the participant, *sensor file* is the name of the sensor files corresponding to the sample, *starts* are the starting indices of the gesture sample for the three sensors — accelerometer, gravity, and gyroscope. Likewise, *ends* are the closing indices for the sample, *label* describes the experiment the sample belongs to. *Label IDs* are the class indices used internally by NOVA. This dataframe is used to obtain a multi-hot encoded dataframe because of the different levels of stratification in the dataset. For example the *tilt* sample in Table 5 belongs to the level 1 "*free*" strata and to the level 2 *sit*ting body posture.

Table 5: Extract from a participant's intermediate dataframe.

User	Sensor file	Starts	Ends	Label	Label IDs
kh026	kh026/trim_kh026_free02	[385, 384, 384]	[511, 509, 509]	[tilt, free, sit]	[1, 2, 5]
kh026	kh026/trim_kh026_free02	[602, 601, 601]	[732, 730, 730]	[tilt, free, sit]	[1, 2, 5]
kh026	kh026/trim_kh026_free02	[731, 729, 729]	[849, 846, 846]	[tilt, free, rest]	[1, 2, 1]

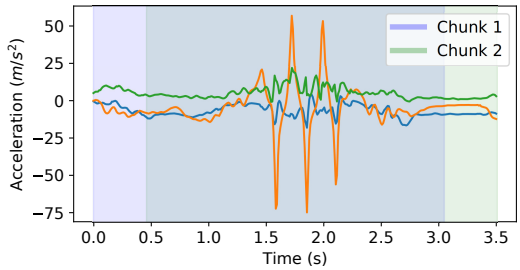
3.2.1 One-vs-all Classification Task

The classification task is devised as a recognition task, whether the performed gesture or activity was the intended one or not. This is a binary classification task with the positive class being the intended action and negative class being the rejection class. For payment gesture recognition, both the "tilt" and "shake" gestures are merged to form a single payment class. The "everyday" superclass with the "rest" class for gestures is used to form the rejection class. For steering activity recognition, the "driving" class forms the positive class and the rejection class consists of the "passenger" class, the "everyday" superclass, and the "rest" class for activities.

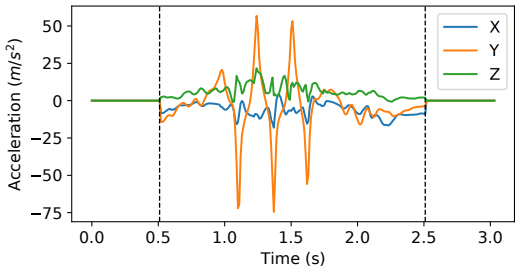
The reason for reducing this multi-class multi-strata classification problem to a binary classification problem is to be able to assess the network's capability for rejecting false positives reliably in sensitive applications like performing payments under a variety of situations. Unintended triggering of payment will have severe impact on the user experience and incur financial losses. Narrowing down this problem in a one-vs-all classifier enables the payment gesture to stand out from unintended everyday activities. This also helps with ensuring the neural network has enough samples and variability per class — in this case two classes, to prevent overfitting.

3.2.2 Time-series

For classification, approximately three seconds were considered appropriate as the maximum duration during which a gesture takes, which has also been done previously for activity recognition by L&H. It is sufficiently long to be able to reliably capture most gestures generated with intentional hand movements. 304 samples at 100 Hz are used which represent ≈ 3.04s worth of IMU sensor data. This arbitrary length of 304 is chosen to be able to generate spectrograms of specific dimensions. Labelled sequences shorter than the length of 304 (≈ 3.04s) are zero-padded on both ends; the longer sequences are split into chunks of length 304 according to the overlapping strategy shown in Figure 21. These chunks are then fed to the network as inputs for training and inference.



(a) Strategy for creating chunks from a sample longer than 3.04 s.



(b) Symmetric zero-padding for samples shorter than 3.04 s.

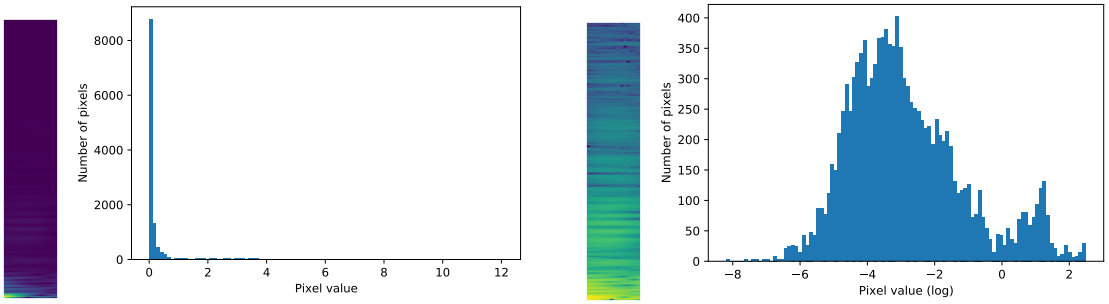
Figure 21: Padding and splitting strategies for fixing the input size for use in CNNs.

3.2.3 Spectrograms

Spectrograms help in visualising the evolution of frequencies in a waveform over time. This is achieved by applying a Short-Time Fourier Transform (STFT) with a sliding window function over the waveform of interest. Spectrograms have found extensive use in the field of audio signal processing and can be a useful tool to capture the periodic nature of human activities.

The 3.04 s long time-series data sequences prepared in Section 3.2.2 are used to generate spectrograms. A Hanning window function was used for the STFT along with an FFT length of 64, yielding a total of 33 frequency bins for a resolvable frequency range of 0-50 Hz from a signal sampled at 100 Hz, which gives a frequency resolution of ≈ 1.5 Hz. The STFT was calculated without padding to minimise edge artifacts in the resulting spectrogram. A sliding window overlap of 75% was chosen to have sufficient resolution along the temporal axis to minimise data redundancy and unnecessary additional compute STFT operations [17]. This yields 16 rolling frames for a 304 sample long signal which creates a spectrogram of dimensions 33×16. The highest frequency bin for the range of 48.5–50 Hz is discarded to conform the dimensions to powers of 2 for an easier handling of maxpooling operations in the deep learning network. The spectrogram matrix consists mostly of smaller values with sparse high values in regions of interest making the pixel distribution of the

spectrogram heavily skewed and similar to that of a power function as shown in Figure 22. So, the computed spectrogram is scaled from the amplitude scale to the logarithmic (dB) scale with the aim of Gaussianising the pixel distribution within a single spectrogram to enable faster convergence of the network during training [91].



(a) Linearly scaled spectrogram.

(b) Log scaled spectrogram.

Figure 22: Pixel distributions for spectrogram values under different scaling and their visual differences. Both spectrograms were generated from the same signal.

3.3 Neural network design

Recurrent networks find common use in applications of machine learning on sequential data of arbitrary lengths. Mobile SDKs provide native implementations on the smartwatch for common layers and operations, like vanilla Recurrent Neural Network (RNN), CNNs, pooling, normalisations, and activations like Rectified Linear Unit (ReLU) and TanH. However, vanilla RNNs tend to suffer from the problem of vanishing gradients over long sequences of data. Recurrent network operations also tend to be less parallelisable because of their sequential nature, making them computationally slower for training. CNNs offer a solution around these problems with the caveat of having a predetermined fixed-sized input. This enables us to use CNNs for gestures, which tend to be of short durations, by fixing a threshold duration (in our case, approximately 3 seconds). This also has the added benefit that CNN based classifiers are easily reusable for transfer learning as feature extractors for other domains or for subsequent use with recurrent networks with the reduced latent representations.

We consider the binary classification task of whether a triggered gesture or activity denotes the positive label (payment for gesture recognition, steering for activity recognition). Variations of simple CNN-based neural network architectures are designed and proposed with the aim of deployment in ready-to-use application-oriented scenarios on the low-end hardware of smartwatches. To keep the preprocessing overhead to an absolute minimum, the “raw” sensor data from the accelerometer, gyroscope, and the gravity sensors is used as it comes from the smartwatch API.

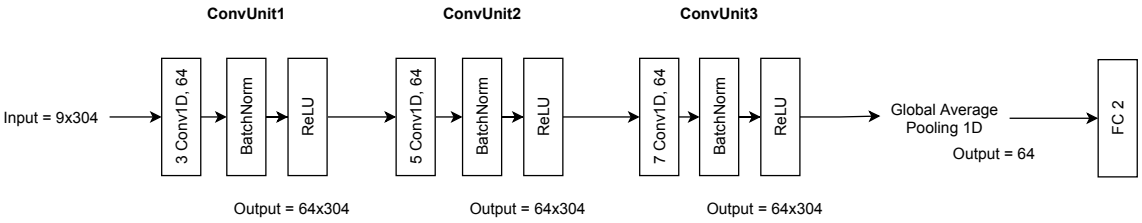
3.3.1 Network inputs and CNN features

The proposed base model's priority is to work on the raw sensor data for recognition. For the model input, 304 time-series samples representing ≈ 3.04s of IMU data from each of three sensors (accelerometer, gyroscope, and gravity sensor) having 3-axis measurements are used. These contribute 3 channels (X, Y, Z) per sensor, resulting in a total of 9 channelled input data sequence making the input shape 9×304. For the time-series model, this is passed through 3 Convolutional Units (ConvUnits) where each ConvUnit is defined as a block consisting of a 1D padded convolutional operation, a 1D batch normalisation, followed by the ReLU activation function. The design for the ConvUnit is similar to that of a conventional convolutional block of the ResNet but extending its 2D operations for the 1D sensor signals in our case of time-series input. The proposed architecture variation utilising the time-series data is shown in Figure 23a.

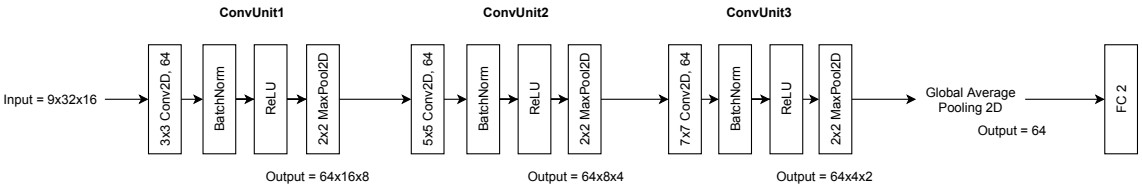
This network architecture is extended to 2-dimensions for spectrogram inputs, in order to compare the proposed approach to Laput and Harrison's, which represents the current state-of-the-art. The 9-channel 304 sample long time-series is converted channel-wise into spectrograms. Each axis of the three sensors results in one spectrogram yielding a net input shape of $9 \times 32 \times 16$. To reduce dimensionality across the 2D ConvUnits, a 2×2 maxpooling is incorporated but the overall recipe remains the same as shown in Figure 23b. This architecture is further expanded by building and training end-to-end a mid-level fusion network with a shared classifier and inputs as both the time-series and spectrogram modalities. The motivation of using both modalities is that the resulting network would be able benefit from the information in both the time and frequency domains as they are correlated [100]. This network, shown in Figure 23c consists of a time-series head and a spectrogram head which are the CNN feature extractors from the previously discussed architectures.

3.3.2 CNN feature classification

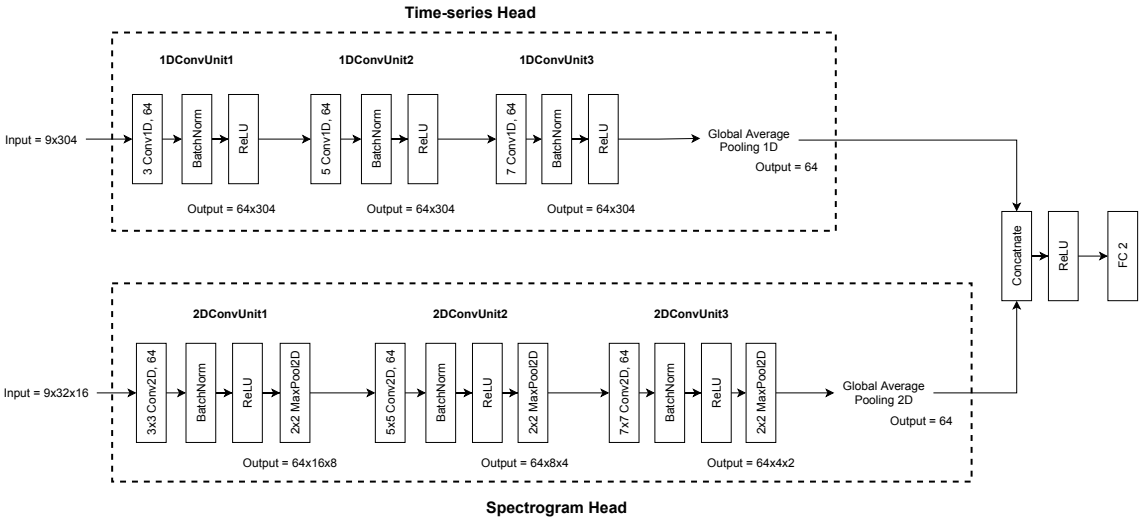
The use of CNN features has been popularised primarily in transfer learning. This enables cross-domain learning which can be useful when faced with the issue of having a smaller dataset. Features learnt from a larger dataset can be reused to help with and fine-tune to the new domain. Each ConvUnit has 64 kernels for feature extraction and the size of kernels increases from 3 to 5 to 7 with each successive ConvUnit to view the signal at different receptive fields. The output of the final ConvUnit is then passed to a Global Average Pooling (GAP) layer which collapses information along the temporal dimension, with the idea of forcing the ConvUnits to not work just as feature extractors but to output feature scores to help the final classification task in the subsequent layers. GAP also reduces the number of learnable parameters in the network helping to reduce the model size substantially which minimises the risk of overfitting when compared to the conventional VGG16-like approach of flattening the convolutional features as a vector and passing them to fully-connected layers directly; we also get around the need for using dropout for regularisation because the fully-connected layers are small enough to avoid overfitting on the low-dimensional dataset used for training. Finally, the pooled tensor is passed through a fully-connected layer



(a) 1D convolution based model architecture for time-series inputs.



(b) 2D convolution based model architecture for spectrogram inputs model.



(c) Modality fusion model for both time-series and spectrogram inputs.

Figure 23: Network architectures

which gives the class scores.

3.4 Extending to Activity Recognition

The same network used for payment gesture recognition can be used for activity recognition by training on the steering data. The activity signals for driving tend to be longer than the fixed window of 3.04s and so are split into multiple 3.04s using the splitting strategy discussed in Section 3.2.2. Inference is performed only on a 3.04s sample of steering. This is called the "one-shot" detection for steering activity. For a more comprehensive, on-the-fly approach, this method can be further extended to run on multiple incoming 3.04s samples and the inferences can be pooled and a majority vote or moving average performed to obtain a more concrete prediction over longer durations as shown in Equation 20 where f represents the neural network,

$\mathbf{x_t}^{(3.04s)}$ is the input sample of 3.04 s duration at time t, and N is the number of samples to be considered for pooling. Figure 24 depicts this process of voting.

$$y = \frac{1}{N} \sum_{t=0}^{N} f\left(\mathbf{x_t}^{(3.04s)}\right) \tag{20}$$

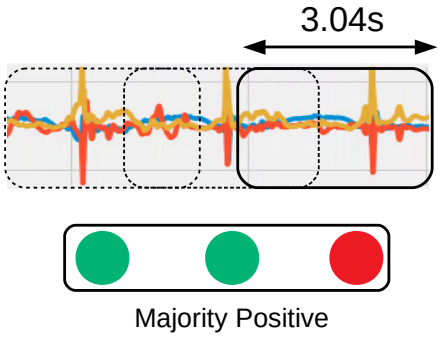


Figure 24: The process of majority voting for accommodating longer sensor samples for activity recognition for the current model designed for 3.04 s duration input.

3.5 Augmentation strategies

3.5.1 Time-series

The time series samples are subjected to random augmentations, each with probability $p = 0.5$. These augmentation strategies aim to artificially inflate the dataset and produce pseudo-realistic perturbations to the raw samples. The factors are chosen on the conservative end to minimise the effects of distribution shifts when compared to the clean data [101]. The augmentations are described below and presented in Figure 26.

- **Random time shift:** The signal is shifted across the time axis by a minimum change factor of 1% and a maximum factor of 10% of the sample duration in either direction. The factor is chosen randomly from an underlying uniform distribution between the specified limits. So, a factor of +5% means the sample is shifted by 5% of its length in the positive direction. Equation 22 depicts the shifting process for an input sample x_t shifted by Δt, sampled from a uniform distribution U, yielding y_t.

$$\Delta t \sim U(-0.1, -0.01) \cup U(0.01, 0.1) \tag{21}$$

$$y_t = x_{t+\Delta t} \tag{22}$$

- **Random frequency shift:** Librosa [102] is used to process the time series sensor signal as an audio signal. The sample is subjected to a time stretch

function to compress or dilate the signal. The minimum change factor is 0% and the maximum factor is set to 3% of the sample duration and the value is drawn from a uniform distribution. A factor of -1% means that the signal is now shorter by 1% (thereby increasing the motion frequency).

- **Random amplification:** The values in the sample are multiplied by a random value between 98% and 102% of the original value as shown in Equation 24. No changes in sample length occur in this transform.

$$A \sim U(0.98, 1.02) \tag{23}$$
$$y_t = Ax_t \tag{24}$$

- **Random axis rotation:** The accelerometer and gravity sensor channels of the sample are rotated along the X-axis by an angle between ±3°. The gyroscope follows a local coordinate system and does not need to be rotated since it outputs values in a relative frame of reference unlike accelerations. Figure 25 shows the proposed coordinate augmentation for the sensor signals. Rotation along X-axis can be represented by the rotation matrix in Equation 26.

$$\theta \sim U(-3°, 3°) \tag{25}$$
$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{26}$$
$$\mathbf{a}' = R\mathbf{a} = R \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \tag{27}$$

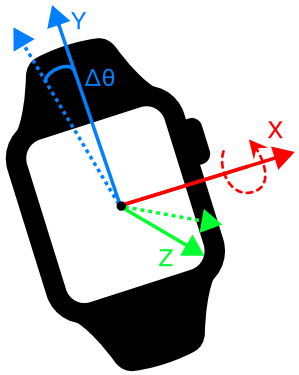


Figure 25: Random axis rotation augmentation for the sensor signals on the local coordinate system of the smartwatch.

- **Random noise:** A Gaussian modelled noise is added to the sensor sample. The noise distribution parameters are estimated by taking extended samples from a region of no activity in the participant data and is then subjected to a sixth-order Butterworth filtering process with a cut-off frequency at 25 Hz as shown in Figure 27. The difference in the filtered and the aligned raw signal then yields the error signal and thus an estimate of the noise parameters for modelling the Gaussian noise. This is repeated for all sensors — accelerometer, gravity, gyroscope, to obtain three Gaussian models for each sensor. The distributions are $\mathcal{N}(0, 0.125)$, $\mathcal{N}(0, 0.015)$, and $\mathcal{N}(0, 0.028)$ for accelerometer, gravity, and gyroscope respectively. Equation 29 represents noise addition to a single channel of accelerometer.

$$n \sim \mathcal{N}(0, 0.0125) \tag{28}$$

$$a'_x = a_x + n \tag{29}$$

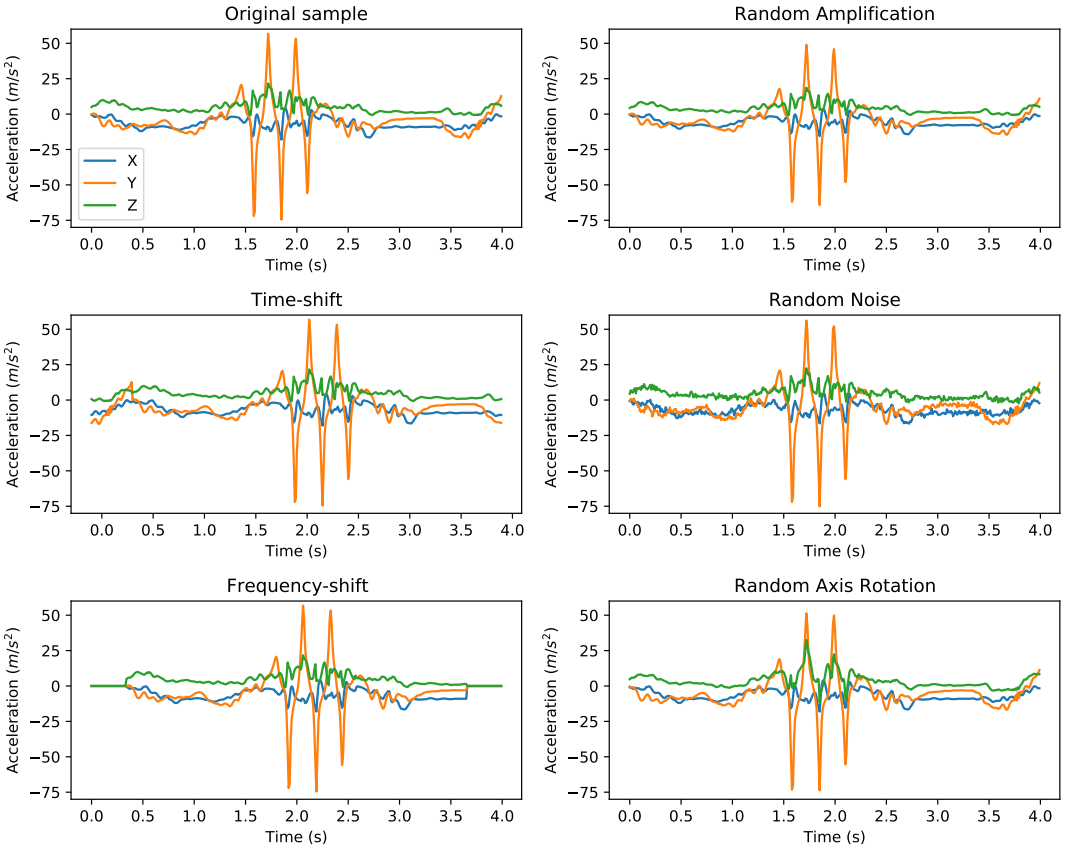


Figure 26: Original time-series sample and the possible augmentations applied to it. The effects are exaggerated for visualisation.

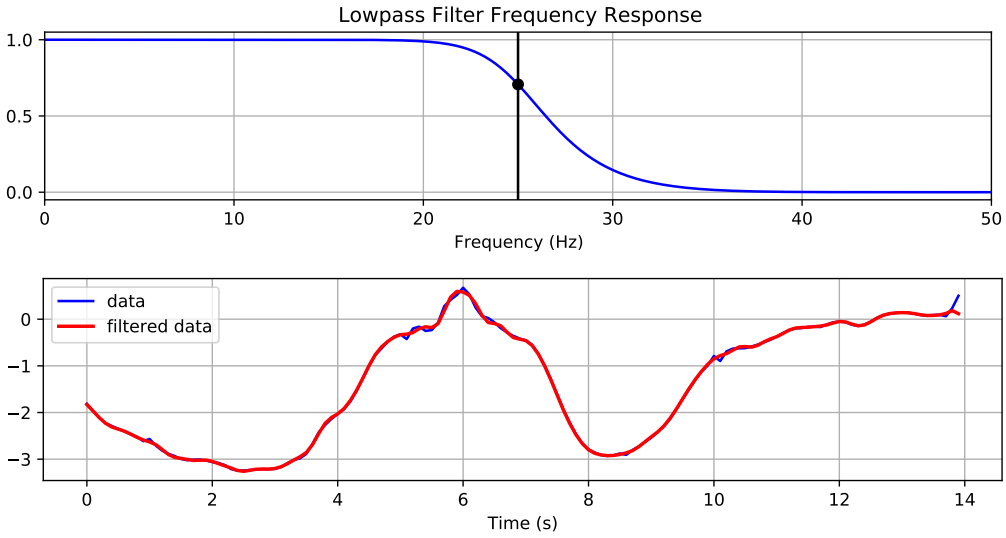


Figure 27: Frequency response of the sixth-order Butterworth filter and the filtering process to estimate sensor noise parameters. Average and standard deviation of the difference/error signal gives the mean and standard deviation of the Gaussian.

3.5.2 Spectrograms

Spectrogram augmentations can be achieved by two means. One is to augment the original sensor or audio signal first and then generate spectrograms and the other is to augment the spectrogram itself as shown by Nanni et al [57]. This thesis follows the latter approach to ensure independence of the two modalities and incorporates known augmentation strategies like SpecAugment [95] which have shown promising results for speech recognition. The augmentations applied are discussed below and are shown in Figure 28

- **SpecAugment:** Random patch erasing along the frequency and temporal axes is performed with a mask width upper bound of 10% of the spectrogram dimensions. The mask width is drawn from a uniform distribution and one mask is applied each to the temporal and frequency axes. The mask type is set to "mean" which replaces the value within the mask with the mean channel value.

- **Random amplification:** The same protocol is followed as that for the time-series sample except for spectrograms, the factor is additive and not multiplicative due to the logarithmic scale of values as shown in Equation 31.

$$A \sim U(0.98, 1.02) \tag{30}$$

$$y_t = x_t + \log A \tag{31}$$

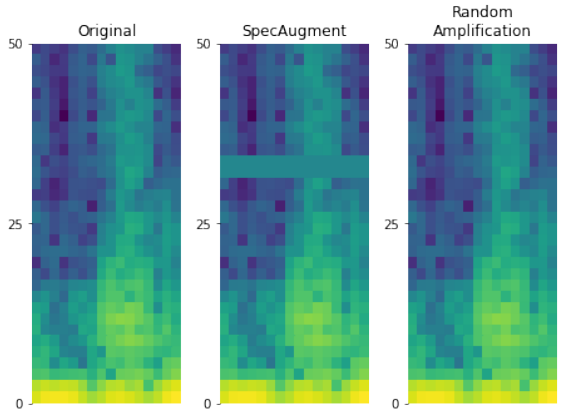


Figure 28: Original spectrogram sample and the augmentations applied to it.

3.6 Quantifying robustness

We have looked at several interpretations of robustness in terms of boundary smoothness, optimisation landscape smoothness, and robustness to adversarial inputs. However, at a more application-oriented level, robustness in user-experience can be defined and perceived as an application or service's reliability of reproducing the same result under different individuals and conditions. The dataset in Section 3.1 divides the data classes further into per-user basis, as well as under varying situations. This enables the ability to partition the data into user-specific and condition-specific chunks for applying a modified k-fold cross-validation resulting in leave-one-user-out (LOUO) and leave-one-condition-out (LOCO) cross validations.

The LOUO has 24 folds for the 24 participants in the dataset. For each fold, a model is trained on 23 participants and the remaining participant is used for cross-validation. Similarly, LOCO has 6 folds for 6 situations in both gesture and activity cross-validation. The payment gesture recognition setup has three situations for strap tightness and three of body posture, while steering activity has three situations for strap tightness and three for driving condition. These cross-validations help to gather model behaviours extensively for the entire combination of users and situations. The metrics for all the folds are logged and the mean metric defines the average model behaviour while the standard deviation of the metrics justifies the model's robustness to how an average model performs to an unseen user or situation not in the training set.This method of evaluation helps to narrow down the particular situations or individuals for which the model performs poorly.

Further, the models trained using LOUO can be evaluated on the left-out users in a stratified manner. This means that instead of using the classes for the left-out users as one test set, the test set is separated into the separate conditions to paint a clearer picture of how the model performs for an unseen user under different situations. This evaluation is coined as the *situational robustness* of the model for the new user.

4 Experiments

The proposed model variations are compared with an extensive test-bench reflective of real life scenarios when the model is subject to deployment. We consider the binary classification task of intended action recognition.

4.1 Experimental setup

Each network is trained for 60 epochs on batches of size 128 of 3.04s samples with the Adam optimiser using an initial learning rate of 0.05 and an exponential learning rate scheduler ($\gamma = 0.9$). A random weighted-sampler is used to remove bias against minority sample classes. The general categorical cross-entropy loss and two logits for the binary classification task instead of the conventional binary cross-entropy loss (BCELoss) with a single logit. All networks were trained on a Dell Precision 5820 Ubuntu 20.04 workstation with Intel Xeon W-2133 (12-core 1.2-3.9GHz) and an NVIDIA TU106 (GeForce RTX 2070). PyTorch 1.8.0 is used as the deep learning framework along with Tensorboard for monitoring model evolution.

4.2 Cross-validations

Accuracy has been traditionally used as a standard metric for almost all classification tasks. But for imbalanced datasets or for multiclass problems, accuracy can be a misleading metric. To address this, we report metrics like precision, recall, the macro-F1 score, and the balanced accuracy which take into account these issues. This is especially important for the cross-validation methods used in the experiments like leave-one-user-out and leave-one-condition-out since a simple accuracy would suppress the minority class.

We consider both the mean of the metrics and their standard deviations to be important indicators during our evaluation. A high mean metric value is prioritised but if two approaches happen to be tied for the same place then we consider the one with the lower standard deviation to be the better approach.

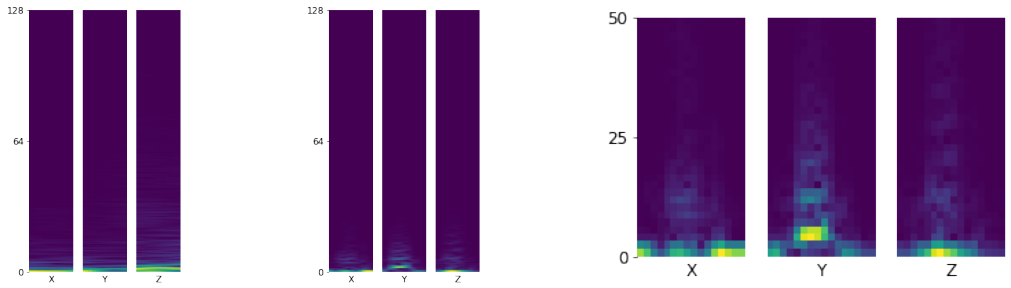
4.3 Baseline comparison

For a state-of-the-art baseline classifier, the model proposed by Laput and Harrison (L&H) is used for comparison. The L&H model takes a 3-channel spectrogram as input of dimensions 256×48 from the accelerometer sensor. As mentioned in the previous section, we change the final layer to output for 2 classes instead of their 25 class for our gesture recognition task.

The L&H dataset is not directly usable on our proposed architectures. This is because the L&H dataset is provided in the form of 3-channel spectrograms without the original raw time-series data. Naive downsampling of their spectrograms for use with our model would not be representative of real-world testing since spectrograms have both temporal and frequency axes which cannot be scaled independently. Access to the raw time-series data would have enabled us to generate spectrograms with

our parameters (Section 3.2.3). However, the parameters they used in their study can be inferred and their procedure replicated.

The L&H spectrogram generation process for L&H is performed by upsampling the time-series sequences from 100 Hz to 4 kHz. A cubic interpolation is performed on the time-series signal and the signal is evaluated at an effective sampling rate of 4 kHz. Then their steps for spectrogram generation are recreated with an FFT window size of 4096, a 2.998 s sample interpolated at 4 kHz, and a hop size of 168, yielding a 2049 × 48 sized spectrogram. Only the bottom 256 bins are kept representing frequencies from 0–128 Hz at a resolution of 0.5 Hz. We did not have access to the high sampling rate of 4 kHz for our data collection. Note that due to the property of the Nyquist sampling rate, it is fundamentally impossible to resample a 100 Hz signal to 4 kHz by the Fourier method [103]. However, we have tried to circumvent this problem with interpolation. The spectograms obtained with this method were visually consistent to those in the original study as can be seen in Figure 29.



(a) 256 × 48 activity spectrogram from L&H dataset [16].

(b) 256 × 48 spectrogram from 4kHz interpolated gesture signal.

(c) 32 × 16 spectrogram from 100 Hz gesture signal for the 2DConv-based network architecture.

Figure 29: Accelerometer spectrograms for the Laput and Harrison baselines and for the proposed 2DConv-based architecture. (b) and (c) were generated from the same gesture signal.

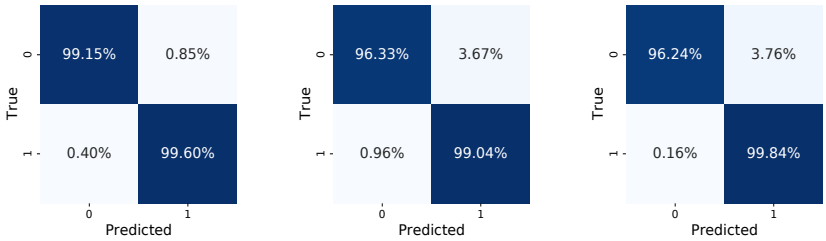
5 Results and Discussion

5.1 Payment Gesture Recognition

Figure 30 summarises the results of the payment gesture cross-validation experiments for robustness. The result summary is presented as a collection of confusion matrices to help examine the classifier performance at a glance, with class 0 as the positive class and class 1 as the rejection or negative class.

5.1.1 User-independent experiments

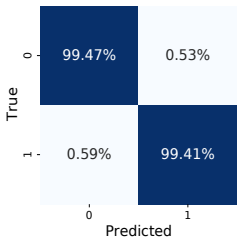
Following the conventional approach for training a deep learning classifier for unpartitioned data, we pool together all participant (or user) data and collapse all levels of situations (strap tightness, body posture) under their respective superclasses of payment and rejection. The resulting singular dataset contains samples from all



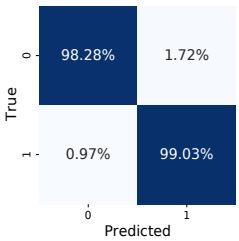
(a) User-independent

(b) Net leave-user-out

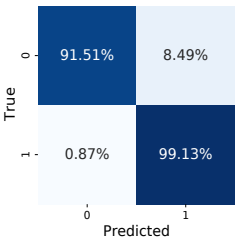
(c) Net leave-condition-out



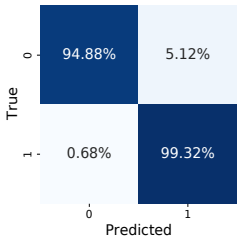
(d) Tight strap



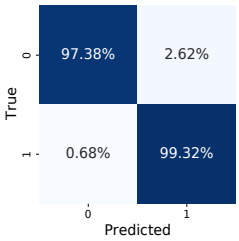
(e) Loose strap



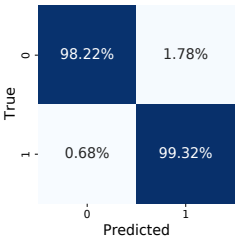
(f) Free condition



(g) Sitting



(h) Standing



(i) Walking

Figure 30: Confusion matrices for the payment gesture binary classification experiments for the modality fusion network architecture. If there are multiple folds in the experiment, then the matrices are averaged across all folds to generate a "net" confusion matrix.

users and situations. The dataset is randomly shuffled and split into 75% training, 12.5% validation, and 12.5% testing. This experiment aims to demonstrate the model's capability to classify new samples from a user who is already present in the training data. Table 6 shows the results for the experiment for the different model architectures.

5.1.2 Leave-one-user-out experiments

A classifier, after deployment, seldom is retrained on fresh data. For a classification task involving participants, it is not always possible to train on a sample size representative of the target population. Hence, ensuring robustness across users becomes paramount to mitigate variability that may arise due to individual differences in performing gestures. To demonstrate how a model (pretrained on a set of users) performs when used by a new unseen user (not present in the training data), we perform the leave-one-user-out cross-validation. For each cross-validation fold, data from one user is held out for testing and the remaining participants are used for

Table 6: Binary classification performance on the test set for payment gesture detection on the user-independent experiments (averaged over 5 independent training runs).

Model Architecture	Accuracy	Balanced Acc	Macro-F1	Precision	Recall
Baseline L&H ($n_{\text{class}} = 2$)	0.994 ± 0.002	0.993 ± 0.002	0.993 ± 0.002	0.992 ± 0.003	0.993 ± 0.001
Time-series (1DConv)	0.992 ± 0.001	0.991 ± 0.001	0.990 ± 0.001	0.991 ± 0.001	0.990 ± 0.002
Time-series (+ Augment)	0.991 ± 0.001	0.990 ± 0.001	0.990 ± 0.001	0.991 ± 0.001	0.988 ± 0.001
Spectrogram (2DConv)	0.992 ± 0.001	0.991 ± 0.001	0.991 ± 0.001	0.990 ± 0.002	0.992 ± 0.001
Spectrogram (+ Augment)	0.993 ± 0.002	0.992 ± 0.002	0.992 ± 0.002	0.991 ± 0.002	0.992 ± 0.002
Modality fusion (Time -series + Spectrogram)	$\mathbf{0.995 \pm 0.001}$	$\mathbf{0.994 \pm 0.001}$	$\mathbf{0.994 \pm 0.001}$	$\mathbf{0.994 \pm 0.001}$	$\mathbf{0.995 \pm 0.001}$
Modality fusion (+ Augment)	0.994 ± 0.000	0.993 ± 0.000	0.993 ± 0.000	0.992 ± 0.001	0.994 ± 0.001

training the classifier. The modality fusion network performs the best on all fronts for an unseen user as shown in Table 7 with the extra information it gets from both the temporal and frequency domains.

Table 7: Classification performance on unseen users for payment gesture detection in the leave-user-out experiments ($n_{\text{fold}} = 24$).

Model Architecture	Accuracy	Balanced Acc	Macro-F1	Precision	Recall
Baseline L&H ($n_{\text{class}} = 2$)	0.979 ± 0.016	0.976 ± 0.018	0.975 ± 0.019	0.972 ± 0.024	0.980 ± 0.014
Time-series (1DConv)	0.976 ± 0.018	0.974 ± 0.018	0.973 ± 0.019	0.973 ± 0.021	0.974 ± 0.019
Time-series (+ Augment)	0.975 ± 0.020	0.972 ± 0.022	0.971 ± 0.023	0.970 ± 0.024	0.974 ± 0.023
Spectrogram (2DConv)	0.976 ± 0.022	0.973 ± 0.023	0.972 ± 0.025	0.969 ± 0.024	0.977 ± 0.024
Spectrogram (+ Augment)	0.974 ± 0.027	0.971 ± 0.028	0.970 ± 0.030	0.967 ± 0.028	0.974 ± 0.031
Modality fusion (Time -series + Spectrogram)	$\mathbf{0.982 \pm 0.016}$	$\mathbf{0.979 \pm 0.018}$	$\mathbf{0.979 \pm 0.019}$	$\mathbf{0.977 \pm 0.024}$	$\mathbf{0.982 \pm 0.016}$
Modality fusion(+ Augment)	0.979 ± 0.022	0.977 ± 0.023	0.976 ± 0.025	0.974 ± 0.026	0.979 ± 0.025

5.1.3 Situational robustness experiments

We design an experiment similar to the leave-one-user-out where the test dataset for the left-out user is split into the multiple constituent situations. For the payment gesture, those situations are for watch tightness (tight, loose, free), and body stature (sit, stand, walk). The pre-trained network from the leave-one-user-out experiment is then evaluated against these split datasets and the results are totalled across all user folds for these six situations. Table 8 shows the modality fusion model is in the lead again with higher mean metrics and smaller standard deviations in most situations. Our spectrogram model ranks next at similar but marginally better performance to the baseline L&H.

5.1.4 Leave-one-condition-out experiments

Finally, we study the scenario of having a missing situation in the training set. To demonstrate the robustness of the models against unseen conditions, we design a leave-group-out experiment for the different situations in the dataset. We refer to

Table 8: Comparison of the classification performance of the models for specific situations for an unseen user ($n_{\text{fold}} = 24$).

Model Architecture	Metric	Tight strap	Loose strap	Free condition	Sitting	Standing	Walking
Baseline L&H ($n_{\text{class}} = 2$)	Accuracy	0.989 ± 0.013	0.985 ± 0.015	0.973 ± 0.028	0.974 ± 0.028	0.980 ± 0.021	0.986 ± 0.013
	Balanced Acc.	0.982 ± 0.019	0.976 ± 0.022	0.956 ± 0.038	0.965 ± 0.031	0.972 ± 0.023	0.980 ± 0.018
	Macro-F1	0.981 ± 0.021	0.976 ± 0.023	0.953 ± 0.045	0.963 ± 0.034	0.971 ± 0.025	0.979 ± 0.019
Time-series (1DConv)	Accuracy	0.973 ± 0.019	0.980 ± 0.022	$\mathbf{0.976 \pm 0.021}$	0.980 ± 0.013	$\mathbf{0.987 \pm 0.011}$	0.985 ± 0.011
	Balanced Acc.	0.957 ± 0.028	0.969 ± 0.034	$\mathbf{0.967 \pm 0.021}$	0.971 ± 0.019	0.980 ± 0.016	0.978 ± 0.016
	Macro-F1	0.955 ± 0.032	0.968 ± 0.036	$\mathbf{0.965 \pm 0.023}$	0.971 ± 0.020	0.979 ± 0.017	0.978 ± 0.017
Time-series (+ Augment)	Accuracy	0.986 ± 0.012	0.981 ± 0.021	0.970 ± 0.028	0.970 ± 0.041	0.979 ± 0.016	0.984 ± 0.014
	Balanced Acc.	0.979 ± 0.017	0.971 ± 0.021	0.950 ± 0.048	0.961 ± 0.036	0.969 ± 0.024	0.977 ± 0.021
	Macro-F1	0.978 ± 0.018	0.968 ± 0.036	$\mathbf{0.965 \pm 0.023}$	0.971 ± 0.020	0.979 ± 0.017	0.978 ± 0.017
Spectrogram (2DConv)	Accuracy	0.990 ± 0.013	0.985 ± 0.018	0.972 ± 0.029	0.978 ± 0.022	0.982 ± 0.017	0.987 ± 0.015
	Balanced Acc.	0.984 ± 0.020	0.976 ± 0.027	0.956 ± 0.042	0.970 ± 0.026	0.975 ± 0.022	0.980 ± 0.021
	Macro-F1	0.983 ± 0.022	0.975 ± 0.029	0.951 ± 0.050	0.968 ± 0.029	0.974 ± 0.024	0.979 ± 0.023
Spectrogram (+ Augment)	Accuracy	0.989 ± 0.018	0.986 ± 0.018	0.969 ± 0.038	0.976 ± 0.024	0.981 ± 0.023	0.988 ± 0.014
	Balanced Acc.	0.982 ± 0.027	0.979 ± 0.025	0.952 ± 0.051	0.966 ± 0.029	0.974 ± 0.027	0.982 ± 0.020
	Macro-F1	0.981 ± 0.028	0.978 ± 0.027	0.947 ± 0.060	0.964 ± 0.031	0.973 ± 0.028	0.982 ± 0.021
Modality fusion (Time -series + Spectrogram)	Accuracy	$\mathbf{0.994 \pm 0.006}$	$\mathbf{0.989 \pm 0.009}$	0.976 ± 0.033	$\mathbf{0.981 \pm 0.025}$	0.987 ± 0.023	$\mathbf{0.991 \pm 0.007}$
	Balanced Acc.	$\mathbf{0.990 \pm 0.010}$	$\mathbf{0.982 \pm 0.014}$	0.963 ± 0.042	$\mathbf{0.974 \pm 0.026}$	$\mathbf{0.982 \pm 0.022}$	$\mathbf{0.987 \pm 0.011}$
	Macro-F1	$\mathbf{0.990 \pm 0.011}$	$\mathbf{0.982 \pm 0.014}$	0.958 ± 0.053	$\mathbf{0.973 \pm 0.029}$	$\mathbf{0.981 \pm 0.025}$	$\mathbf{0.986 \pm 0.011}$
Modality fusion (+ Augment)	Accuracy	0.993 ± 0.008	0.988 ± 0.014	0.972 ± 0.036	0.980 ± 0.027	0.984 ± 0.023	0.990 ± 0.010
	Balanced Acc.	0.988 ± 0.014	0.981 ± 0.020	0.957 ± 0.047	0.973 ± 0.027	0.978 ± 0.024	0.985 ± 0.015
	Macro-F1	0.988 ± 0.014	0.980 ± 0.022	0.952 ± 0.058	0.972 ± 0.030	0.976 ± 0.026	0.985 ± 0.015

this as the leave-condition-out experiment. For the payment scenario, we have six situations but they are not fully independent. The first level of the variation in the payment dataset is the watch strap tightness while the second level is the body posture. However, variation in watch tightness and body posture occurs together. So, the experiment is split into two parts for a total of six leave-condition-out folds as follows.

- **Watch tightness:** The dataset has the body posture situations collapsed and merged giving a three-situation dataset of watch strap tight, loose, and free. A leave-group-out cross-validation is performed on these three situations, e.g., train on tight and loose and validate on free. This part yields three folds for the experiment.
- **Body posture:** Similarly, the watch strap tightness situations are collapsed which gives a dataset for sitting, standing, and walking. This, again, yields three leave-group-out folds for the experiment. As shown in Table 9, the modality fusion model outperforms all the other approaches.

5.2 Steering Activity Recognition

The same CNN-based models are trained for activity recognition on 3.04 s samples for a "one-shot" detection of steering recognition. The positive class consists of the steering activity while the negative class is the combination of "passenger", "everyday", and "rest". Likewise, the same cross-validation methods are used to measure the effectiveness and performance of the models in an activity recognition setup. The training and evaluation methodology is the same as that of gesture recognition in Section 5.1 unless stated otherwise. Figure 31 summarises the cross-validation results for the "one-shot" steering recognition.

Table 9: All-user classification performance on unseen situations in the leave-condition-out experiments ($n_{\text{fold}} = 6$).

Model Architecture	Accuracy	Balanced Acc	Macro-F1	Precision	Recall
Baseline L&H ($n_{\text{class}} = 2$)	0.991 ± 0.011	0.985 ± 0.019	0.984 ± 0.020	0.976 ± 0.030	0.994 ± 0.008
Time-series (1DConv)	0.989 ± 0.010	0.982 ± 0.017	0.981 ± 0.018	0.976 ± 0.029	0.987 ± 0.007
Time-series (+ Augment)	0.989 ± 0.009	0.982 ± 0.016	0.982 ± 0.016	0.977 ± 0.025	0.988 ± 0.008
Spectrogram (2DCOnv)	0.990 ± 0.014	0.983 ± 0.024	0.982 ± 0.027	0.975 ± 0.039	0.991 ± 0.010
Spectrogram (+ Augment)	0.989 ± 0.014	0.982 ± 0.024	0.981 ± 0.026	0.974 ± 0.038	0.990 ± 0.011
Modality fusion (Time -series + Spectrogram)	$\mathbf{0.992 \pm 0.011}$	$\mathbf{0.987 \pm 0.018}$	0.986 ± 0.020	0.981 ± 0.030	$\mathbf{0.992 \pm 0.030}$
Modality fusion (+ Augment)	0.991 ± 0.011	0.986 ± 0.018	$\mathbf{0.986 \pm 0.019}$	$\mathbf{0.981 \pm 0.028}$	0.991 ± 0.008

5.2.1 User-independent experiments

The dataset is randomly shuffled and split into 75% training, 12.5% validation, and 12.5% testing. The user-independent results are summarised in Table 10. Augmentations seem to help improve the average performance of the recogniser in the traditional three-way hold-out experimental setup.

Table 10: Binary classification performance on the test set for steering activity recognition on the user-independent experiments (averaged over 5 independent runs).

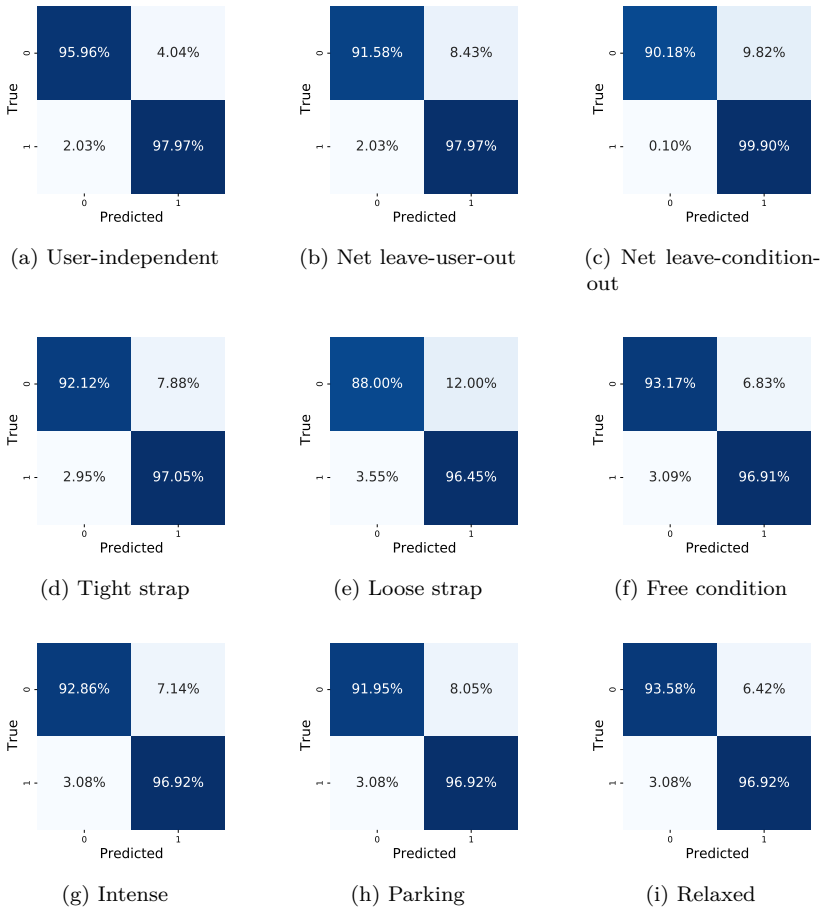
Model Architecture	Accuracy	Balanced Acc	Macro-F1	Precision	Recall
Baseline L&H ($n_{\text{class}} = 2$)	0.922 ± 0.006	0.909 ± 0.006	0.909 ± 0.006	0.909 ± 0.009	0.910 ± 0.009
Time-series (1DConv)	0.936 ± 0.006	0.930 ± 0.006	0.929 ± 0.006	0.935 ± 0.006	0.924 ± 0.007
Time-series (+ Augment)	0.942 ± 0.003	0.932 ± 0.004	0.931 ± 0.004	0.938 ± 0.004	0.926 ± 0.004
Spectrogram (2DConv)	0.954 ± 0.003	0.947 ± 0.003	0.947 ± 0.003	0.947 ± 0.003	0.947 ± 0.004
Spectrogram (+ Augment)	0.958 ± 0.001	0.951 ± 0.001	0.951 ± 0.001	0.949 ± 0.003	0.952 ± 0.001
Modality fusion (Time -series + Spectrogram)	0.964 ± 0.004	0.957 ± 0.005	0.957 ± 0.005	0.958 ± 0.003	0.956 ± 0.006
Modality fusion (+ Augment)	$\mathbf{0.973 \pm 0.002}$	$\mathbf{0.969 \pm 0.003}$	$\mathbf{0.969 \pm 0.003}$	$\mathbf{0.970 \pm 0.004}$	$\mathbf{0.969 \pm 0.003}$

5.2.2 Leave-one-user-out experiments

Leave-one-user-out experiment results for steering recognition are summarised in Table 11. Augmentations appear to slightly hurt the robustness of the recogniser in a user-invariability test. However for spectrograms, the robustness has improved with a slight trade-off in recognition accuracy. Spectrograms appear to be the better modality than the raw sensor data for activity recognition when it comes to individual robustness.

5.2.3 Situational robustness experiments

Robustness of steering recognition under varying watch conditions and driving situations is presented in Table 12. Augmentations seem to slightly improve the



(a) User-independent (b) Net leave-user-out (c) Net leave-condition-out

(d) Tight strap (e) Loose strap (f) Free condition

(g) Intense (h) Parking (i) Relaxed

Figure 31: Confusion matrices for the steering activity binary classification experiments for the modality fusion network architecture. If there are multiple folds in the experiment, then the matrices are averaged across all folds to generate a "net" confusion matrix.

recogniser performance and robustness across several situations for the unseen user but the effect is marginal.

5.2.4 Leave-one-condition-out experiments

Following the same experimental methodology as in payment gesture recognition, the performance of the models when subjected to new unseen conditions is discussed in Table 13. Augmentations tend to have minimal impact on the conditional robustness of the classifier. The modality and the corresponding network architecture appear to play a larger role in affecting the recogniser robustness. Spectrogram modality again outperforms the raw time-series for activity recognition. This was not the case in gesture recognition where both performed equally well in almost all cross-validations.

Table 11: Classification performance on unseen users for steering activity recognition in the leave-user-out experiments ($n_{\text{fold}} = 24$).

Model Architecture	Accuracy	Balanced Acc	Macro-F1	Precision	Recall
Baseline L&H ($n_{\text{class}} = 2$)	0.882 ± 0.042	0.867 ± 0.044	0.863 ± 0.046	0.867 ± 0.046	0.867 ± 0.046
Time-series (1DConv)	0.897 ± 0.045	0.884 ± 0.054	0.881 ± 0.055	0.891 ± 0.057	0.877 ± 0.057
Time-series (+ Augment)	0.895 ± 0.047	0.881 ± 0.055	0.878 ± 0.057	0.888 ± 0.056	0.875 ± 0.059
Spectrogram (2DConv)	0.936 ± 0.027	0.926 ± 0.027	0.924 ± 0.029	0.924 ± 0.028	0.929 ± 0.032
Spectrogram (+ Augment)	0.935 ± 0.024	0.925 ± 0.026	0.924 ± 0.027	0.924 ± 0.028	0.926 ± 0.028
Modality fusion (Time -series + Spectrogram)	0.950 ± 0.024	0.942 ± 0.027	0.942 ± 0.028	0.941 ± 0.031	$\mathbf{0.944 \pm 0.027}$
Modality fusion (+ Augment)	$\mathbf{0.951 \pm 0.024}$	$\mathbf{0.944 \pm 0.026}$	$\mathbf{0.942 \pm 0.027}$	$\mathbf{0.943 \pm 0.028}$	0.944 ± 0.028

Table 12: Comparison of the classification performance of the models for specific situations for an unseen user ($n_{\text{fold}} = 24$).

Model Architecture	Metric	Tight strap	Loose strap	Free condition	Intense	Parking	Relaxed
Baseline L&H ($n_{\text{class}} = 2$)	Accuracy	0.903 ± 0.045	0.893 ± 0.050	0.896 ± 0.047	0.905 ± 0.042	0.899 ± 0.046	0.904 ± 0.041
	Balanced Acc.	0.831 ± 0.092	0.792 ± 0.128	0.841 ± 0.052	0.821 ± 0.115	0.812 ± 0.092	0.846 ± 0.055
	Macro-F1	0.819 ± 0.093	0.779 ± 0.126	0.829 ± 0.059	0.809 ± 0.114	0.809 ± 0.094	0.829 ± 0.064
Time-series(1DConv)	Accuracy	0.919 ± 0.038	0.901 ± 0.039	0.913 ± 0.039	0.919 ± 0.034	0.921 ± 0.034	0.911 ± 0.041
	Balanced Acc.	0.852 ± 0.111	0.792 ± 0.131	0.863 ± 0.070	0.836 ± 0.119	0.860 ± 0.095	0.844 ± 0.111
	Macro-F1	0.844 ± 0.111	0.785 ± 0.130	0.853 ± 0.071	0.827 ± 0.118	0.850 ± 0.096	0.830 ± 0.099
Time-series(+ Augment)	Accuracy	0.917 ± 0.043	0.902 ± 0.038	0.910 ± 0.039	0.919 ± 0.035	0.921 ± 0.035	0.911 ± 0.040
	Balanced Acc.	0.850 ± 0.116	0.797 ± 0.131	0.857 ± 0.067	0.836 ± 0.119	0.859 ± 0.096	0.844 ± 0.088
	Macro-F1	0.841 ± 0.116	0.789 ± 0.130	0.847 ± 0.070	0.826 ± 0.118	0.849 ± 0.098	0.829 ± 0.098
Spectrogram (2DConv)	Accuracy	0.953 ± 0.028	0.942 ± 0.033	0.949 ± 0.032	0.954 ± 0.030	0.949 ± 0.029	0.954 ± 0.032
	Balanced Acc.	0.900 ± 0.093	0.847 ± 0.142	0.916 ± 0.044	0.886 ± 0.013	0.890 ± 0.093	0.917 ± 0.047
	Macro-F1	0.895 ± 0.095	0.842 ± 0.142	0.910 ± 0.050	0.881 ± 0.128	0.885 ± 0.095	0.910 ± 0.055
Spectrogram (+ Augment)	Accuracy	0.952 ± 0.024	0.943 ± 0.030	0.947 ± 0.028	0.951 ± 0.026	0.948 ± 0.025	0.954 ± 0.029
	Balanced Acc.	0.895 ± 0.091	0.847 ± 0.141	0.910 ± 0.040	0.879 ± 0.125	0.887 ± 0.091	0.914 ± 0.050
	Macro-F1	0.891 ± 0.092	0.843 ± 0.141	0.906 ± 0.045	0.875 ± 0.125	0.882 ± 0.092	0.908 ± 0.055
Modality fusion (Time -series + Spectrogram)	Accuracy	0.965 ± 0.020	0.954 ± 0.026	0.964 ± 0.022	$\mathbf{0.965 \pm 0.023}$	0.964 ± 0.020	0.966 ± 0.042
	Balanced Acc.	0.917 ± 0.094	0.863 ± 0.146	0.938 ± 0.037	$\mathbf{0.900 \pm 0.129}$	0.915 ± 0.094	0.934 ± 0.042
	Macro-F1	0.914 ± 0.095	0.859 ± 0.014	0.935 ± 0.041	$\mathbf{0.897 \pm 0.130}$	0.911 ± 0.095	0.930 ± 0.048
Modality Fusion (+ Augment)	Accuracy	$\mathbf{0.966 \pm 0.021}$	$\mathbf{0.956 \pm 0.028}$	$\mathbf{0.964 \pm 0.022}$	0.965 ± 0.024	$\mathbf{0.966 \pm 0.020}$	$\mathbf{0.967 \pm 0.023}$
	Balanced Acc.	$\mathbf{0.919 \pm 0.096}$	$\mathbf{0.866 \pm 0.147}$	$\mathbf{0.939 \pm 0.033}$	0.899 ± 0.128	$\mathbf{0.920 \pm 0.094}$	$\mathbf{0.935 \pm 0.043}$
	Macro-F1	$\mathbf{0.916 \pm 0.097}$	$\mathbf{0.862 \pm 0.147}$	$\mathbf{0.936 \pm 0.037}$	0.896 ± 0.129	$\mathbf{0.917 \pm 0.096}$	$\mathbf{0.930 \pm 0.051}$

5.3 Model comparison

In Table 14, we compare the model sizes, their footprint, and their number of parameters to gauge the deployability of these architectures on the low-end hardware of wearable devices. It can be seen that the proposed models not only are highly accurate and robust to both situational and individual variations but also require minimal computational resources, which is desirable for deployment on smartwatches.

5.4 Discussion

Prior literature, the experimental methods, and the presented results together help to answer the research questions formulated in Section 1.

1. How does one define robustness in general?
   Robustness, in general, is defined as the stability of an algorithm to produce consistent results to perturbed inputs. This perturbation can be due to variation in the environment, in the process of data collection, processing, noise, or

Table 13: All-user classification performance on unseen situations in the leave-condition-out experiments ($n_{\text{fold}} = 6$).

Model Architecture	**Accuracy**	**Balanced Acc**	**Macro-F1**	**Precision**	**Recall**
Baseline L&H ($n_{\text{class}} = 2$)	0.971 ± 0.005	0.941 ± 0.011	0.937 ± 0.012	0.906 ± 0.016	0.976 ± 0.011
Time-series (1DConv)	0.961 ± 0.014	0.922 ± 0.032	0.921 ± 0.033	0.923 ± 0.044	0.921 ± 0.020
Time-series (+ Augment)	0.958 ± 0.013	0.916 ± 0.031	0.916 ± 0.032	0.921 ± 0.044	0.912 ± 0.020
Spectrogram (2DConv)	0.981 ± 0.008	0.962 ± 0.017	0.960 ± 0.019	0.938 ± 0.026	0.986 ± 0.008
Spectrogram (+ Augment)	0.980 ± 0.008	0.961 ± 0.016	0.958 ± 0.018	0.935 ± 0.025	0.986 ± 0.008
Modality fusion (Time -series + Spectrogram)	$\mathbf{0.985 \pm 0.007}$	$\mathbf{0.970 \pm 0.016}$	$\mathbf{0.969 \pm 0.017}$	$\mathbf{0.952 \pm 0.024}$	$\mathbf{0.989 \pm 0.010}$
Modality fusion (+ Augment)	0.985 ± 0.010	0.969 ± 0.022	0.968 ± 0.023	0.950 ± 0.032	0.988 ± 0.011

Table 14: Comparison of the properties of network architectures and their inputs.

Model Architecture	**Sensor sample rate** (Hz)	**Input dims**	**Number of parameters**	**Size** (MB)	**Training time** (s)	**Total MFLOPs**
Baseline L&H ($n_{\text{class}} = 2$)	4000	$[3, 256, 8]$	$23,667,062$	91	825	1907.66
Time-series(1DConv)	**100**	$[\mathbf{9}, \mathbf{304}]$	**51,394**	**0.2**	**130**	**31.16**
Spectrogram (2DConv)	**100**	$[9, 32, 16]$	$308,802$	1.2	178	44.56
Modality fusion (Time-series + Spectrogram)	**100**	$[9, 304]$, $[9, 32, 16]$	$360,194$	1.2	197	59.90

intentional interference. Robustness in an algorithm or model allows for some leeway and assurance to the user that they will not obtain a wildly different output for a slightly out-of-distribution input.

2. What does robustness mean in the context of user experience?
   In the context of user experience, robustness can be broadly categorised into two components — situational, and individual. Situational robustness is the algorithmic stability of a model to be invariant to changes in the environment or situation. This may or may not be under the direct control of the user. If the same user is displaced from one environment to another then a situationally robust algorithm should be stable in both environments.

   On the other hand, individual robustness is the model's stability to be invariant to change in the user themselves. If the user of the device is changed and the new user performs a similar input command then an individually robust algorithm will have stable output for both the new and old users.

3. How does one quantify robustness in this context?
   Robustness in the context of user experience can be quantified by evaluation the model or algorithm using different variants of cross-validation methods which fix one part of the input and cycle through the rest to iteratively generate a outputs for all the possible relevant conditions. Performing a statistical analysis of the model performance under these situations then portrays the robustness of the model. For example, a perfectly individually robust model would be invariant to change in users and would produce the same output for same or similar input from all users.

4. What are the factors that contribute to robustness of a deep learning model?
   From prior research and our empirical analysis, the model size and complexity, the type of operations, initial state of the optimiser, model hyperparameters, and quality of the training dataset are all contributing factors to the robustness of a DL model. Some of these factors appear to affect robustness more than the rest as is seen in our experiments where model complexity and input modality was a more contributing factor than the preconceived notion of an improved distribution from the augmented input.

5. How to train a computationally light deep learning model and measure its robustness in user-experience?
   For deployment, an application-oriented method of design should be followed to approach a low computational complexity. One such approach is to train a difficult-to-overfit classifier which is presented in this thesis to ensure the model does not overfit on the training data resulting in poor inference performance. Use of model pruning, and compression techniques can also be used to further lighten the model. Further, the deployed network is to be trained on all the available training data but for evaluating and measuring its robustness, it should be trained in the computationally intensive way of performing cross-validations to ensure that the model can perform reasonably well in all real-world scenarios.

6 Conclusions

The thesis shows that robust and deployable DL models for gesture and activity recognition are possible for devices with low-end hardware like smartwatches. A cross-validation type of method to evaluate individual and situational variability in a multi-strata dataset and helps to quantify user-perceived robustness. Further, designing appropriate variations of common neural network architectures plays a major role in improving the performance and the robustness of gesture recognition. Augmentation strategies can help improve the performance of the classifier in certain situations, particularly in activity recognition where it really shines, but it is less of a determining factor when targeting robustness for HAR.

The proposed models achieve upwards of 95% on challenging recognition tasks across different users and situations. The dataset used guarantees a rich data distribution which is exploited by architectural design to mitigate overfitting — a pressing problem on smaller, low-dimensional datasets, improving generalisability over the state-of-the-art. Moreover, the impact of using correlated features or modalities is demonstrated to increase the performance and robustness of the recogniser even further. The payment gesture recogniser using a multi-modal approach dwarves the state-of-the-art in an application-oriented non-academic setup in both performance and robustness. The same is also demonstrated to be true for activity recognition.

From a deployability point-of-view, the proposed architectures are shown to require almost two orders of magnitude less compute operations than the previous state-of-the-art while improving the performance at best and no loss in performance at worst. While prior methods required hacking the device to obtain higher sensor sampling rates, our approach works on out-of-the-box OS APIs which favours less power consumption. The small size guarantees minimal preprocessing requirements and offers huge savings in terms of computational resources and more importantly, batter life in wearables enabling the recogniser to be run fully “offline” without being tethered to a smartphone to offload the recognition task. This also opens the door for these architectures to be further fine-tuned with model pruning or compression.

The proposed methods of robustness quantification and evaluation, along with the model architectures provide a minimum viable solution for gesture and activity recognition and also help to narrow down the specific cases where the recogniser fails frequently. In future work, the simpler task of binary recognition can be extended to multi-class classification. Owing to the small size of our model, it opens opportunities to use the model as a one-vs-all (OVA) classifier unit which can then be extended to multi-class problems with ensembling methods. On-device learning or fine-tuning and transfer learning utilising the CNN layers of the proposed models are also exciting directions for future research in expanding the approach to domains like instrumentation and controls. We are positive that improvements in the dataset, for example, more participants, strata, and in-the-wild collection of data can further improve the benchmark results of the proposed methods.

References

[1] Y. Lai, "A Comparison of Traditional Machine Learning and Deep Learning in Image Recognition", *Journal of Physics: Conference Series*, vol. 1314, p. 012 148, Oct. 2019, ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/1314/1/012148.

[2] P. Morales, "Smart Movement Detection for Android Phones", Universitat Politècnica de Catalunya, Barcelona, Oct. 2016, 60 pp.

[3] F. Demrozi, G. Pravadelli, A. Bihorac, and P. Rashidi, "Human Activity Recognition using Inertial, Physiological and Environmental Sensors: A Comprehensive Survey", *IEEE Access*, vol. 8, pp. 210 816–210 836, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3037715. arXiv: 2004.08821.

[4] Y. Wang, H. Cheng, and M. Q. H. Meng. (Sep. 18, 2020). Pedestrian Motion Tracking by Using Inertial Sensors on the Smartphone. arXiv: 2009.08824 [cs].

[5] T. Michel, P. Geneves, H. Fourati, and N. Layaida, "On attitude estimation with smartphones", in *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Kona, HI: IEEE, Mar. 2017, pp. 267–275, ISBN: 978-1-5090-4327-9. DOI: 10.1109/PERCOM.2017.7917873.

[6] A. Bayat, M. Pomplun, and D. A. Tran, "A Study on Human Activity Recognition Using Accelerometer Data from Smartphones", *Procedia Computer Science*, vol. 34, pp. 450–457, 2014, ISSN: 18770509. DOI: 10.1016/j.procs.2014.07.009.

[7] S. Mahmud, M. T. H. Tonmoy, and K. K. Bhaumik, "Human Activity Recognition from Wearable Sensor Data Using Self-Attention", *Santiago de Compostela*, p. 8, 2020.

[8] A. Jordao, A. C. Nazare Jr., J. Sena, and W. R. Schwartz. (Feb. 1, 2019). Human Activity Recognition Based on Wearable Sensor Data: A Standardization of the State-of-the-Art. arXiv: 1806.05226 [cs], [Online]. Available: http://arxiv.org/abs/1806.05226 (visited on 2021-01-25).

[9] J. Davila, A.-M. Cretu, and M. Zaremba, "Wearable Sensor Data Classification for Human Activity Recognition Based on an Iterative Learning Framework", *Sensors*, vol. 17, no. 6, p. 1287, Jun. 7, 2017, ISSN: 1424-8220. DOI: 10.3390/s17061287.

[10] O. Kerr, "Real-time motion classification from every day activity using a single wearable IMU", p. 70, 2010.

[11] F. J. Ordóñez and D. Roggen, "Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition", p. 25, 2016.

[12] Y. Mohammad, K. Matsumoto, and K. Hoashi, "Primitive activity recognition from short sequences of sensory data", *Applied Intelligence*, vol. 48, no. 10, pp. 3748–3761, Oct. 2018, ISSN: 0924-669X, 1573-7497. DOI: 10.1007/s10489-018-1166-6.

[13] G. Laput, R. Xiao, and C. Harrison, "ViBand: High-Fidelity Bio-Acoustic Sensing Using Commodity Smartwatch Accelerometers", in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, Tokyo Japan: ACM, Oct. 16, 2016, pp. 321–333, ISBN: 978-1-4503-4189-9. DOI: 10.1145/2984511.2984582.

[14] H. Wen, J. Ramos Rojas, and A. K. Dey, "Serendipity: Finger Gesture Recognition using an Off-the-Shelf Smartwatch", in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, San Jose California USA: ACM, May 7, 2016, pp. 3847–3851, ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858466.

[15] A. Soro, G. Brunner, S. Tanner, and R. Wattenhofer, "Recognition and Repetition Counting for Complex Physical Exercises with Deep Learning", p. 22, 2019.

[16] G. Laput and C. Harrison, "Sensing Fine-Grained Hand Activity with Smartwatches", in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, Glasgow Scotland UK: ACM, May 2, 2019, pp. 1–13, ISBN: 978-1-4503-5970-2. DOI: 10.1145/3290605.3300568.

[17] A. Tobola, F. J. Streit, C. Espig, O. Korpok, C. Sauter, N. Lang, B. Schmitz, C. Hofmann, M. Struck, C. Weigand, H. Leutheuser, B. M. Eskofier, and G. Fischer, "Sampling rate impact on energy consumption of biomedical signal processing systems", in *2015 IEEE 12th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, Cambridge, MA, USA: IEEE, Jun. 2015, pp. 1–6, ISBN: 978-1-4673-7201-5. DOI: 10.1109/BSN.2015.7299392.

[18] F. J. González-Cañete and E. Casilari, "Consumption Analysis of Smartphone based Fall Detection Systems with Multiple External Wireless Sensors", *Sensors*, vol. 20, no. 3, p. 622, Jan. 22, 2020, ISSN: 1424-8220. DOI: 10.3390/s20030622.

[19] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015.

[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library", in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035.

[21] S. Branco, A. G. Ferreira, and J. Cabral, "Machine Learning in Resource-Scarce Embedded Systems, FPGAs, and End-Devices: A Survey", *Electronics*, vol. 8, no. 11, p. 1289, Nov. 5, 2019, ISSN: 2079-9292. DOI: 10.3390/electronics8111289.

[22] Y. Cheng, D. Wang, P. Zhou, and T. Zhang. (Jun. 14, 2020). A Survey of Model Compression and Acceleration for Deep Neural Networks. arXiv: 1710.09282 [cs], [Online]. Available: http://arxiv.org/abs/1710.09282 (visited on 2021-11-10).

[23] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. (Sep. 22, 2016). Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. arXiv: 1609.07061 [cs], [Online]. Available: http://arxiv.org/abs/1609.07061 (visited on 2021-11-10).

[24] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. (Dec. 15, 2017). Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. arXiv: 1712.05877 [cs, stat], [Online]. Available: http://arxiv.org/abs/1712.05877 (visited on 2021-11-10).

[25] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer. (Jun. 21, 2021). A Survey of Quantization Methods for Efficient Neural Network Inference. arXiv: 2103.13630 [cs], [Online]. Available: http://arxiv.org/abs/2103.13630 (visited on 2021-11-10).

[26] S. Hochreiter, "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 06, no. 02, pp. 107–116, Apr. 1998, ISSN: 0218-4885, 1793-6411. DOI: 10.1142/S0218488598000094.

[27] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory", *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1, 1997, ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1997.9.8.1735.

[28] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. (Sep. 2, 2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv: 1406.1078 [cs, stat], [Online]. Available: http://arxiv.org/abs/1406.1078 (visited on 2021-10-27).

[29] S. Chung, J. Lim, K. J. Noh, G. Kim, and H. Jeong, "Sensor Data Acquisition and Multimodal Sensor Fusion for Human Activity Recognition Using Deep Learning", *Sensors*, vol. 19, no. 7, p. 1716, Apr. 10, 2019, ISSN: 1424-8220. DOI: 10.3390/s19071716.

[30] O. Steven Eyobu and D. Han, "Feature Representation and Data Augmentation for Human Activity Classification Based on Wearable IMU Sensor Data Using a Deep LSTM Neural Network", *Sensors*, vol. 18, no. 9, p. 2892, Aug. 31, 2018, ISSN: 1424-8220. DOI: 10.3390/s18092892.

[31] O. Konak, P. Wegner, and B. Arnrich, "IMU-Based Movement Trajectory Heatmaps for Human Activity Recognition", *Sensors*, vol. 20, no. 24, p. 7179, Dec. 15, 2020, ISSN: 1424-8220. DOI: 10.3390/s20247179.

[32] K. Simonyan and A. Zisserman. (Apr. 10, 2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv: 1409.1556 [cs], [Online]. Available: http://arxiv.org/abs/1409.1556 (visited on 2021-05-17).

[33] K. He, X. Zhang, S. Ren, and J. Sun. (Dec. 10, 2015). Deep Residual Learning for Image Recognition. arXiv: 1512.03385 [cs], [Online]. Available: http://arxiv.org/abs/1512.03385 (visited on 2021-09-28).

[34] T. Shermin, S. W. Teng, M. Murshed, G. Lu, F. Sohel, and M. Paul, "Enhanced Transfer Learning with ImageNet Trained Classification Layer", Mar. 25, 2019.

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database", in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL: IEEE, Jun. 2009, pp. 248–255, ISBN: 978-1-4244-3992-8. DOI: 10.1109/CVPR.2009.5206848.

[36] H. Lu, H. Zhang, and A. Nayak. (Jun. 14, 2020). A Deep Neural Network for Audio Classification with a Classifier Attention Mechanism. arXiv: 2006.09815 [cs, eess, stat], [Online]. Available: http://arxiv.org/abs/2006.09815 (visited on 2021-01-25).

[37] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager. (Nov. 16, 2016). Temporal Convolutional Networks for Action Segmentation and Detection. arXiv: 1611.05267 [cs], [Online]. Available: http://arxiv.org/abs/1611.05267 (visited on 2021-05-17).

[38] I. A. Lawal and S. Bano, "Deep Human Activity Recognition Using Wearable Sensors", p. 4, 2019.

[39] K. Yaguchi, K. Ikarigawa, R. Kawasaki, W. Miyazaki, Y. Morikawa, C. Ito, M. Shuzo, and E. Maeda, "Human Activity Recognition Using Multi-input CNN Model with FFT Spectrograms", p. 4, 2020.

[40] A. Bevilacqua, K. MacDonald, A. Rangarej, V. Widjaya, B. Caulfield, and T. Kechadi. (Jun. 5, 2019). Human Activity Recognition with Convolutional Neural Networks. arXiv: 1906.01935 [cs, stat].

[41] O. J. Woodman, "An introduction to inertial navigation", p. 37,

[42] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm", in *2011 IEEE International Conference on Rehabilitation Robotics*, Zurich: IEEE, Jun. 2011, pp. 1–7, ISBN: 978-1-4244-9862-8 978-1-4244-9863-5 978-1-4244-9861-1. DOI: 10.1109/ICORR.2011.5975346.

[43] H. Yan, Q. Shan, and Y. Furukawa, "RIDI: Robust IMU Double Integration", in *Computer Vision – ECCV 2018*, ser. Lecture Notes in Computer Science, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., vol. 11217, Cham: Springer International Publishing, 2018, pp. 641–656, ISBN: 978-3-030-01260-1 978-3-030-01261-8. DOI: 10.1007/978-3-030-01261-8_38.

[44] S. Herath, H. Yan, and Y. Furukawa, "RoNIN: Robust Neural Inertial Navigation in the Wild: Benchmark, Evaluations, & New Methods", in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France: IEEE, May 2020, pp. 3146–3152, ISBN: 978-1-72817-395-5. DOI: 10.1109/ICRA40945.2020.9196860.

[45] T. Michel, P. Genevès, H. Fourati, and N. Layaïda, "Attitude estimation for indoor navigation and augmented reality with smartphones", *Pervasive and Mobile Computing*, vol. 46, pp. 96–121, Jun. 2018, ISSN: 15741192. DOI: 10.1016/j.pmcj.2018.03.004.

[46] T. Michel, H. Fourati, P. Geneves, and N. Layaida, "A comparative analysis of attitude estimation for pedestrian navigation with smartphones", in *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Banff, AB, Canada: IEEE, Oct. 2015, pp. 1–10, ISBN: 978-1-4673-8402-5. DOI: 10.1109/IPIN.2015.7346767.

[47] A. Poulose, O. S. Eyobu, and D. S. Han, "An Indoor Position-Estimation Algorithm Using Smartphone IMU Sensor Data", *IEEE Access*, pp. 1–1, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2891942.

[48] P. Aqueveque, S. Sobarzo, F. Saavedra, C. Maldonado, and B. Gómez, "Android platform for realtime gait tracking using inertial measurement units", *European Journal of Translational Myology*, vol. 26, no. 3, Jul. 6, 2016, ISSN: 2037-7460, 2037-7452. DOI: 10.4081/ejtm.2016.6144.

[49] L. Chen, Y. Zhao, Y. Zhang, S. Fan, and Y. Jia, "An Online Prediction and Trajectory Tracking Method for Human Activity Recognition", in *Proceedings of the 2020 9th International Conference on Computing and Pattern Recognition*, Xiamen China: ACM, Oct. 30, 2020, pp. 18–23, ISBN: 978-1-4503-8783-5. DOI: 10.1145/3436369.3436458.

[50] D. N. Patel, "Miniatured Inertial Motion and Position Tracking and Visualization Systems Using Android Wear Platform", p. 101,

[51] Yun Li, Xiang Chen, Xu Zhang, Kongqiao Wang, and Z. J. Wang, "A Sign-Component-Based Framework for Chinese Sign Language Recognition Using Accelerometer and sEMG Data", *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 10, pp. 2695–2704, Oct. 2012, ISSN: 0018-9294, 1558-2531. DOI: 10.1109/TBME.2012.2190734.

[52] T. Deselaers, D. Keysers, J. Hosang, and H. A. Rowley, "GyroPen: Gyroscopes for Pen-Input With Mobile Phones", *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 2, pp. 263–271, Apr. 2015, ISSN: 2168-2291, 2168-2305. DOI: 10.1109/THMS.2014.2365723.

[53] T. Baltrušaitis, C. Ahuja, and L.-P. Morency. (Aug. 1, 2017). Multimodal Machine Learning: A Survey and Taxonomy. arXiv: 1705.09406 [cs], [Online]. Available: http://arxiv.org/abs/1705.09406 (visited on 2021-05-17).

[54] A. Roitberg, T. Pollert, M. Haurilet, M. Martin, and R. Stiefelhagen, "Analysis of Deep Fusion Strategies for Multi-Modal Gesture Recognition", in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Long Beach, CA, USA: IEEE, Jun. 2019, pp. 198–206, ISBN: 978-1-72812-506-0. DOI: 10.1109/CVPRW.2019.00029.

[55] S. Chen and Q. Jin. (Sep. 4, 2017). Multi-modal Conditional Attention Fusion for Dimensional Emotion Prediction. arXiv: 1709.02251 [cs], [Online]. Available: http://arxiv.org/abs/1709.02251 (visited on 2021-11-11).

[56] A. Zadeh, M. Chen, S. Poria, E. Cambria, and L.-P. Morency. (Jul. 23, 2017). Tensor Fusion Network for Multimodal Sentiment Analysis. arXiv: 1707.07250 [cs], [Online]. Available: http://arxiv.org/abs/1707.07250 (visited on 2021-05-17).

[57] L. Nanni, G. Maguolo, and M. Paci, "Data augmentation approaches for improving animal audio classification", *Ecological Informatics*, vol. 57, p. 101 084, May 2020, ISSN: 15749541. DOI: 10.1016/j.ecoinf.2020.101084.

[58] S. Amiriparian, "Deep representation learning techniques for audio signal processing", Technische Universität München, 2019.

[59] S. Shuvaev, H. Giaffar, and A. A. Koulakov. (Dec. 7, 2017). Representations of Sound in Deep Learning of Audio Features from Music. arXiv: 1712.02898 [cs, eess, q-bio], [Online]. Available: http://arxiv.org/abs/1712.02898 (visited on 2021-11-11).

[60] R. A. Solovyev, M. Vakhrushev, A. Radionov, V. Aliev, and A. A. Shvets. (Oct. 4, 2018). Deep Learning Approaches for Understanding Simple Speech Commands. arXiv: 1810.02364 [cs, eess], [Online]. Available: http://arxiv.org/abs/1810.02364 (visited on 2021-11-11).

[61] H. Lee, Y. Largman, P. Pham, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks", in *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, ser. NIPS'09, Red Hook, NY, USA: Curran Associates Inc., Dec. 7, 2009, pp. 1096–1104, ISBN: 978-1-61567-911-9.

[62] D. Jain, H. Ngo, P. Patel, S. Goodman, L. Findlater, and J. Froehlich, "SoundWatch: Exploring Smartwatch-based Deep Learning Approaches to Support Sound Awareness for Deaf and Hard of Hearing Users", in *The 22nd International ACM SIGACCESS Conference on Computers and Accessibility*, Virtual Event Greece: ACM, Oct. 26, 2020, pp. 1–13, ISBN: 978-1-4503-7103-2. DOI: 10.1145/3373625.3416991.

[63] K. Chen, D. Zhang, L. Yao, B. Guo, Z. Yu, and Y. Liu. (Jan. 22, 2021). Deep Learning for Sensor-based Human Activity Recognition: Overview, Challenges and Opportunities. arXiv: 2001.07416 [cs], [Online]. Available: http://arxiv.org/abs/2001.07416 (visited on 2021-01-26).

[64] M. Müller, *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*, 1st ed. 2015. Cham: Springer International Publishing : Imprint: Springer, 2015, 1 p., ISBN: 978-3-319-21945-5. DOI: 10.1007/978-3-319-21945-5.

[65] J. Calic, *Introduction to DSP*. OpenStax CNX, Sep. 23, 2014, 376 pp.

[66] S. G. Johnson and M. Frigo, "A Modified Split-Radix FFT With Fewer Arithmetic Operations", *IEEE Transactions on Signal Processing*, vol. 55, no. 1, pp. 111–119, Jan. 2007, ISSN: 1053-587X. DOI: 10.1109/TSP.2006.882087.

[67] J. O. Smith, *Spectral Audio Signal Processing*. Center for Computer Research in Music and Acoustics (CCRMA), Music Department, Stanford University, 2011.

[68] S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, and M. Bethge, "Improving robustness against common corruptions by covariate shift adaptation", p. 23,

[69] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo, D. Song, J. Steinhardt, and J. Gilmer. (Aug. 13, 2020). The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization. arXiv: 2006.16241 [cs, stat], [Online]. Available: http://arxiv.org/abs/2006.16241 (visited on 2021-03-08).

[70] G. Ortiz-Jiménez, A. Modas, S.-M. Moosavi-Dezfooli, and P. Frossard, "Redundant features can hurt robustness to distribution shift", p. 8,

[71] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry. (Sep. 9, 2019). Robustness May Be at Odds with Accuracy. arXiv: 1805.12152 [cs, stat], [Online]. Available: http://arxiv.org/abs/1805.12152 (visited on 2021-01-26).

[72] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan. (Jun. 24, 2019). Theoretically Principled Trade-off between Robustness and Accuracy. arXiv: 1901.08573 [cs, stat], [Online]. Available: http://arxiv.org/abs/1901.08573 (visited on 2021-01-26).

[73] A. Raghunathan, S. M. Xie, F. Yang, J. Duchi, and P. Liang. (Jul. 6, 2020). Understanding and Mitigating the Tradeoff Between Robustness and Accuracy. arXiv: 2002.10716 [cs, stat], [Online]. Available: http://arxiv.org/abs/2002.10716 (visited on 2021-01-26).

[74] H. Salman, G. Yang, J. Li, P. Zhang, H. Zhang, I. Razenshteyn, and S. Bubeck. (Jan. 9, 2020). Provably Robust Deep Learning via Adversarially Trained Smoothed Classifiers. arXiv: 1906.04584 [cs, stat], [Online]. Available: http://arxiv.org/abs/1906.04584 (visited on 2021-01-26).

[75] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter. (Jun. 15, 2019). Certified Adversarial Robustness via Randomized Smoothing. arXiv: 1902.02918 [cs, stat], [Online]. Available: http://arxiv.org/abs/1902.02918 (visited on 2021-01-26).

[76] L. Li, M. Weber, X. Xu, L. Rimanic, T. Xie, C. Zhang, and B. Li. (Jun. 9, 2020). Provable Robust Learning Based on Transformation-Specific Smoothing. arXiv: 2002.12398 [cs, stat], [Online]. Available: http://arxiv.org/abs/2002.12398 (visited on 2021-03-08).

[77] J. T. Barron. (Apr. 4, 2019). A General and Adaptive Robust Loss Function. arXiv: 1701.03077 [cs, stat], [Online]. Available: http://arxiv.org/abs/1701.03077 (visited on 2021-02-04).

[78] L. Sadouk, T. Gadi, and E. H. Essoufi, "Robust Loss Function for Deep Learning Regression with Outliers", in *Embedded Systems and Artificial Intelligence*, ser. Advances in Intelligent Systems and Computing, V. Bhateja, S. C. Satapathy, and H. Satori, Eds., vol. 1076, Singapore: Springer Singapore, 2020, pp. 359–368, ISBN: 9789811509469 9789811509476. DOI: 10.1007/978-981-15-0947-6_34.

[79] D. Idnani and J. C. Kao, "Learning Robust Representations with Score Invariant Learning", 2020.

[80] O. Deniz, A. Pedraza, N. Vallez, J. Salido, and G. Bueno, "Robustness to adversarial examples can be improved with overfitting", *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 4, pp. 935–944, Apr. 2020, ISSN: 1868-8071, 1868-808X. DOI: 10.1007/s13042-020-01097-4.

[81] L. Rice, E. Wong, and J. Z. Kolter. (Mar. 4, 2020). Overfitting in adversarially robust deep learning. arXiv: 2002.11569 [cs, stat], [Online]. Available: http://arxiv.org/abs/2002.11569 (visited on 2021-11-12).

[82] S. Yeom, I. Giacomelli, A. Menaged, M. Fredrikson, and S. Jha, "Overfitting, robustness, and malicious algorithms: A study of potential causes of privacy risk in machine learning", *Journal of Computer Security*, vol. 28, no. 1, pp. 35–70, Feb. 4, 2020, ISSN: 18758924, 0926227X. DOI: 10.3233/JCS-191362.

[83] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[84] M. Lin, Q. Chen, and S. Yan. (Mar. 4, 2014). Network In Network. arXiv: 1312.4400 [cs], [Online]. Available: http://arxiv.org/abs/1312.4400 (visited on 2021-05-06).

[85] S. Ioffe and C. Szegedy. (Mar. 2, 2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv: 1502.03167 [cs], [Online]. Available: http://arxiv.org/abs/1502.03167 (visited on 2021-11-12).

[86] J. L. Ba, J. R. Kiros, and G. E. Hinton. (Jul. 21, 2016). Layer Normalization. arXiv: 1607.06450 [cs, stat], [Online]. Available: http://arxiv.org/abs/1607.06450 (visited on 2021-07-06).

[87] Y. Wu and K. He. (Jun. 11, 2018). Group Normalization. arXiv: 1803.08494 [cs], [Online]. Available: http://arxiv.org/abs/1803.08494 (visited on 2021-06-08).

[88] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. (Apr. 14, 2019). How Does Batch Normalization Help Optimization? arXiv: 1805.11604 [cs, stat], [Online]. Available: http://arxiv.org/abs/1805.11604 (visited on 2021-08-25).

[89] G. Chen, P. Chen, Y. Shi, C.-Y. Hsieh, B. Liao, and S. Zhang. (May 14, 2019). Rethinking the Usage of Batch Normalization and Dropout in the Training of Deep Neural Networks. arXiv: 1905.05928 [cs, stat], [Online]. Available: http://arxiv.org/abs/1905.05928 (visited on 2021-08-25).

[90] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning", *Journal of Big Data*, vol. 6, no. 1, p. 60, Dec. 2019, ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0.

[91] A. Cheddad, "On Box-Cox Transformation for Image Normality and Pattern Classification", *IEEE Access*, vol. 8, pp. 154 975–154 983, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3018874.

[92] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random Erasing Data Augmentation", *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, pp. 13 001–13 008, Apr. 3, 2020, ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v34i07.7000.

[93] A. Mikolajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem", in *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, Swinoujście: IEEE, May 2018, pp. 117–122, ISBN: 978-1-5386-6143-7. DOI: 10.1109/IIPHDW.2018.8388338.

[94] V. Sandfort, K. Yan, P. J. Pickhardt, and R. M. Summers, "Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks", *Scientific Reports*, vol. 9, no. 1, p. 16 884, Dec. 2019, ISSN: 2045-2322. DOI: 10.1038/s41598-019-52737-x.

[95] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le. (Dec. 3, 2019). SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. arXiv: 1904.08779 [cs, eess, stat].

[96] J. Wang, Y. Chen, Y. Gu, Y. Xiao, and H. Pan, "SensoryGANs: An Effective Generative Adversarial Framework for Sensor-based Human Activity Recognition", in *2018 International Joint Conference on Neural Networks (IJCNN)*, Rio de Janeiro: IEEE, Jul. 2018, pp. 1–8, ISBN: 978-1-5090-6014-6. DOI: 10.1109/IJCNN.2018.8489106.

[97] S. Raschka. (Nov. 10, 2020). Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. arXiv: 1811.12808 [cs, stat], [Online]. Available: http://arxiv.org/abs/1811.12808 (visited on 2021-11-14).

[98] A. Heimerl, T. Baur, F. Lingenfelser, J. Wagner, and E. Andre, "NOVA - A tool for eXplainable Cooperative Machine Learning", in *2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII)*, Cambridge, United Kingdom: IEEE, Sep. 2019, pp. 109–115, ISBN: 978-1-72813-888-6. DOI: 10.1109/ACII.2019.8925519.

[99] T. Baur, A. Heimerl, F. Lingenfelser, J. Wagner, M. F. Valstar, B. Schuller, and E. André, "eXplainable Cooperative Machine Learning with NOVA", *KI - Künstliche Intelligenz*, vol. 34, no. 2, pp. 143–164, Jun. 2020, ISSN: 0933-1875, 1610-1987. DOI: 10.1007/s13218-020-00632-3.

[100] N. Ay, J. Flack, and D. C. Krakauer, "Robustness and complexity co-constructed in multimodal signalling networks", *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 362, no. 1479, pp. 441–447, Mar. 29, 2007, ISSN: 0962-8436, 1471-2970. DOI: 10.1098/rstb.2006.1971.

[101] Z. He, L. Xie, X. Chen, Y. Zhang, Y. Wang, and Q. Tian. (Nov. 21, 2019). Data Augmentation Revisited: Rethinking the Distribution Gap between Clean and Augmented Data. arXiv: 1909.09148 [cs, stat], [Online]. Available: http://arxiv.org/abs/1909.09148 (visited on 2021-11-15).

[102] B. McFee, V. Lostanlen, A. Metsai, M. McVicar, S. Balke, C. Thomé, C. Raffel, F. Zalkow, A. Malek, Dana, K. Lee, O. Nieto, J. Mason, D. Ellis, E. Battenberg, S. Seyfarth, R. Yamamoto, K. Choi, Viktorandreevichmorozov, J. Moore, R. Bittner, S. Hidaka, Z. Wei, Nullmightybofo, D. Hereñú, F.-R. Stöter, P. Friesch, A. Weiss, M. Vollrath, and T. Kim, *Librosa/librosa: 0.8.0*, version 0.8.0, Zenodo, Jul. 22, 2020. DOI: 10.5281/ZENODO.3955228.

[103] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, 2nd ed. Upper Saddle River, N.J: Prentice Hall, 1999, 870 pp., ISBN: 978-0-13-754920-7.

A Payment Gesture

A.1 Leave-one-user-out

The following plots show LUOU confusion matrices for testing on payment gesture recognition. Each matrix belongs to one of the 24 participants in the dataset.

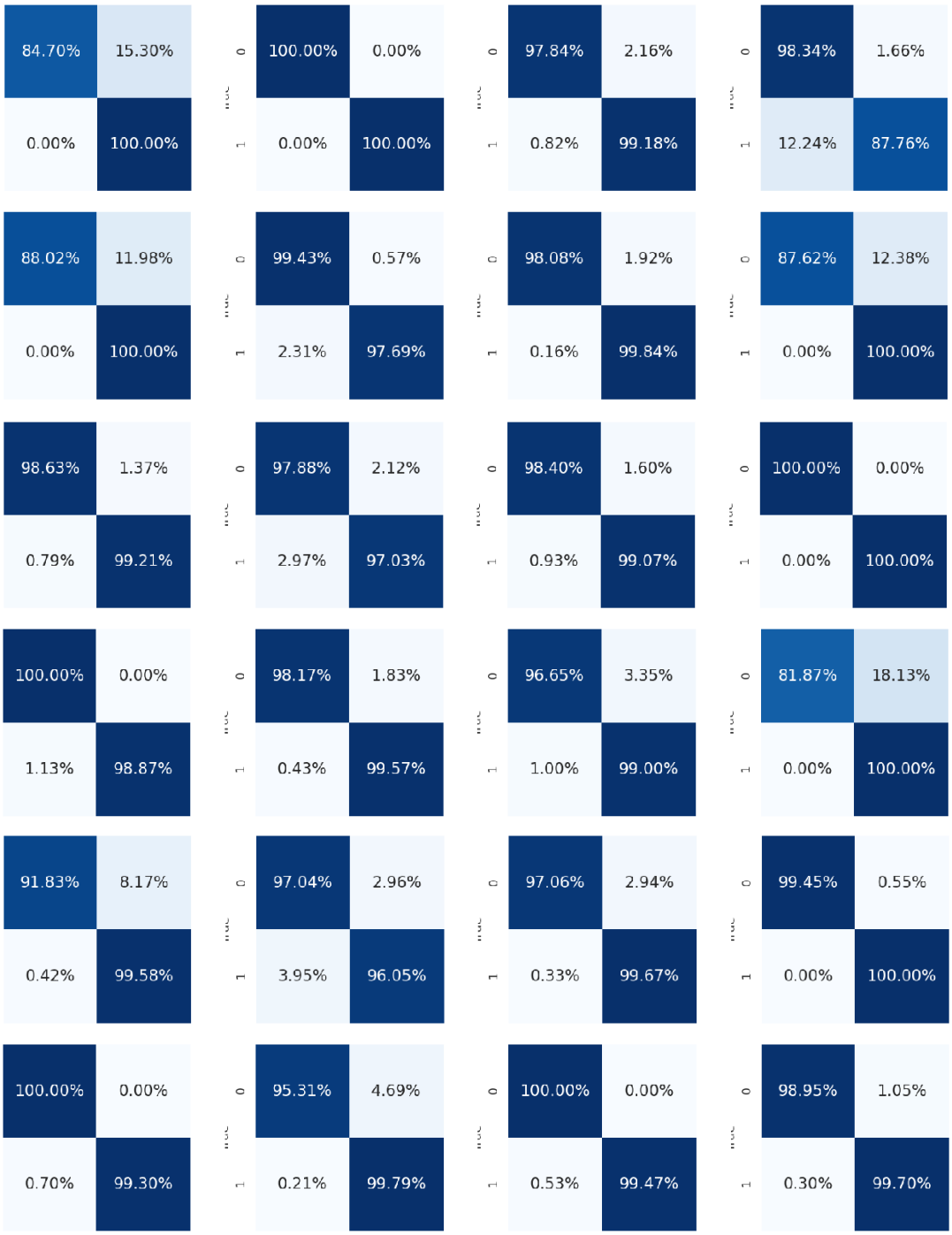


Figure A1: LOUO confusion matrices for all 24 participants for the payment gesture.

B Steering Activity

B.1 Leave-one-user-out

The following plots show LOUO confusion matrices for testing on steering activity recognition. Each matrix belongs to one of the 24 participants in the dataset.

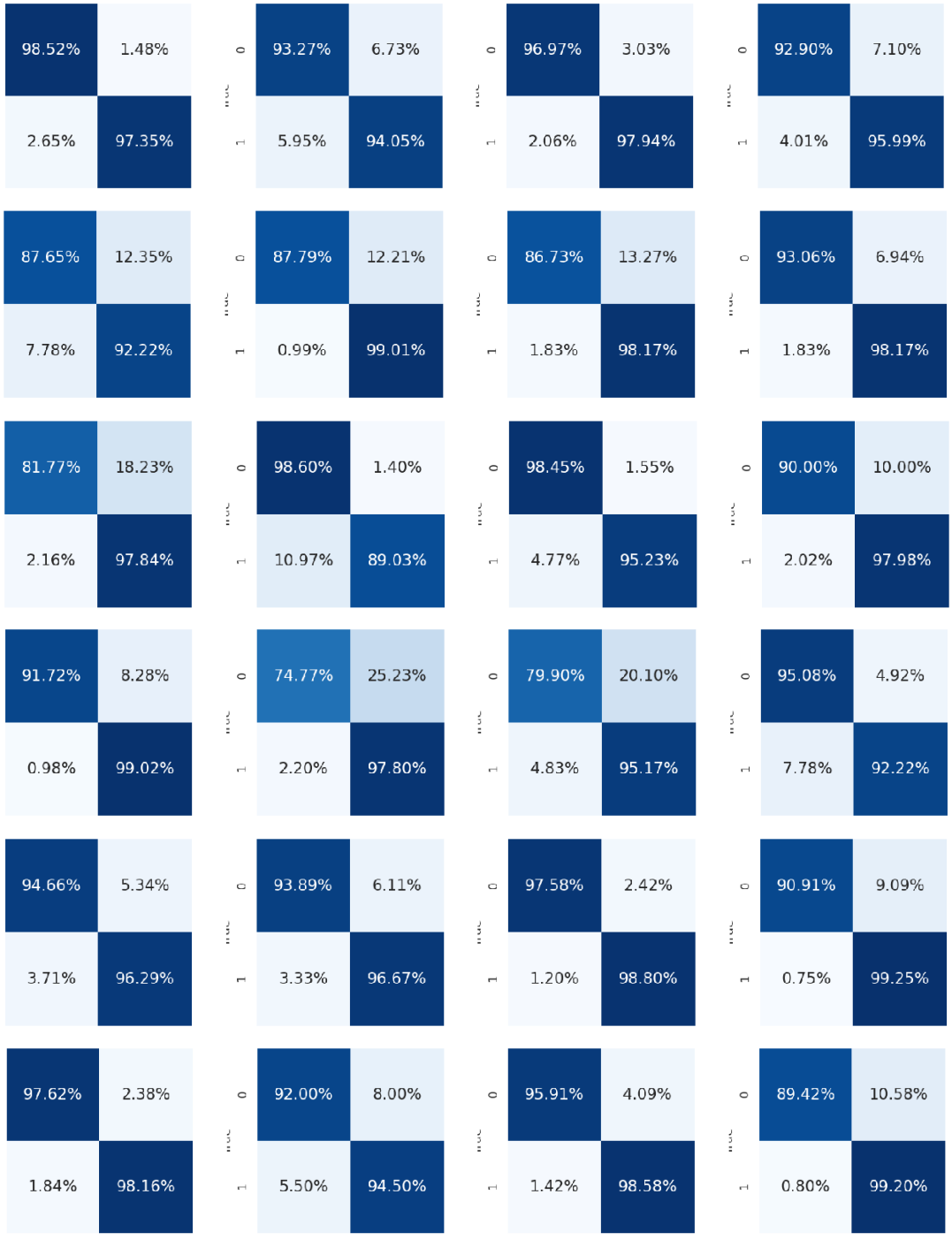


Figure B1: LOUO confusion matrices for all 24 participants for the steering activity.