EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science
Interconnected Resource-aware Intelligent Systems Research Group

Towards domain agnostic Wi-Fi CSI gesture classification

Master's Thesis

Yvan Putra Satyawan

Supervisors:
Prof. Dr. Ir. Nirvana Meratnia
Bram R.D. van Berlo, M.Sc

Eindhoven, September 2023

Abstract

Somewhat ominously, we are getting closer and closer to ubiquitous remote sensing and gesture recognition through the use of only Wi-Fi signals. This is understood as the ability to recognize gestures performed by an individual with nothing more than the interference their bodies cause to surrounding Wi-Fi signals. The main challenge facing the mainstream adoption of such technologies is the lack of generalizability seen in published models against various domain factors. This work aims to bring us closer to such a (dystopian) future where such technologies may be mainstream by presenting DARLInG (Domain Autolabeling through Reinforcement Learning for the Inference of Gestures), a novel approach to domain shift mitigation through the use of domain auto-labeling using reinforcement learning. In this work, we explore DARLInG and show that while promising, further research will be necessary to investigate the true extent of the capabilities of our approach.

Acknowledgements

I would like to express my gratitude to the following:

To my supervisors Nirvana and Bram for giving up their time to guide me, and quite thoroughly, through this thesis,

To my friends, those of you in the DSAI masters with me and our group Belly Dancing as a Service, those of you in Pattern, and to Rani, who helped me in the last few weeks of writing this thesis,

To my family, who supported me throughout the entirety of my masters.

Contents

Contents **vii**

List of Figures **ix**

List of Tables **xiii**

1 Introduction **1**

1.1 Context and Background 2

1.2 Motivation 4

1.3 Problem Statement 5

2 Literature Review **7**

2.1 Wi-Fi for Activity Detection 7

2.2 Wi-Fi CSI for Gesture Recognition 8

2.3 Wi-Fi CSI datasets for Gesture Recognition 10

2.4 Signal-to-Image Transformations 11

2.5 Domain Shift Mitigation Methods 11

2.6 Domain Shift Mitigation using Reinforcement Learning 12

3 Background Knowledge **15**

3.1 Variational Autoencoders 15

3.2 Reinforcement Learning 16

3.2.1 Proximal Policy Optimization 17

3.2.2 Deep Q-Networks 18

3.2.3 Deep Deterministic Policy Gradient 19

3.3 Channel State Information 21

3.4 Body-coordinate Velocity Profile 21

3.5 Signal-to-Image Transformations 22

3.5.1 Gramian Angular Fields 22

3.5.2 Markov Transition Fields . . . 23
3.5.3 Recurrence Plots . . . 23

4 Methodology **25**
4.1 Widar 3.0 . . . 26
4.2 DARLInG . . . 27
4.3 Signal Preprocessing . . . 29
4.3.1 Signal Processing . . . 29
4.3.2 Signal-to-Image Transformation . . . 32
4.4 Signal encoding . . . 34
4.5 Multi-task Learning . . . 36
4.6 Reinforcement Learning Agent . . . 37
4.6.1 Known Domain Encoding . . . 38
4.6.2 Reward Function . . . 38
4.7 Model Training . . . 40

5 Experiments **41**
5.1 Preliminary Experiments . . . 41
5.1.1 Single-Domain Experiments . . . 41
5.1.2 Hyperparameter Optimization . . . 42
5.2 Experimental Setup and Results . . . 44
5.2.1 Extended Run . . . 50

6 Discussion **51**
6.1 Comparison to State-of-the-Art . . . 51
6.2 Research questions . . . 52
6.3 Latent Space and Domain Embedding Analysis . . . 54
6.4 Future Research . . . 56
6.4.1 A Different Architecture . . . 56
6.4.2 Tuning of the RL Agent . . . 56
6.4.3 A Different Training Regime . . . 57

7 Conclusions **59**

Appendix **64**

A Dataset Analysis **65**

B Latent Space and Domain Embedding t-SNE Filtered by User **73**

List of Figures

1.1 A WiZ A60 Tunable White light bulb, one example of an IoT smart device with an integrated Wi-Fi Radio [33]. 1

3.1 A series of slices of a BVP representation of a pushing and pulling gesture, sliced along the time axis. The main velocity component, which represents the movement of the subject's hand, is highlighted with a red circle. This set of images is taken directly from [46]. 22

4.1 An abstracted diagram of the proposed DARLInG architecture. 26
4.2 Typical positioning of the access points and subjects in Widar. The image is sourced from [46]. Loc shows the different torso locations used and orient the different orientations the subject may be at. 26
4.3 The basic paradigm of Reinforcement Learning where an agent interacts with its environment through actions a and receives a new state s and reward r in return. 28
4.4 Details of the signal preprocessing module. The module is comprised of traditional signal preprocessing and a signal-to-image transformation. 28
4.5 An example of the amplitude shift signal after being transformed by each step of the signal processing pipeline. Only the first 600 samples of one channel from one tranceiver link is shown here for illustrative purposes. 29
4.6 An example of the phase shift signal after being transformed by each step of the signal processing pipeline. Only the first 600 samples of one channel from one tranceiver link is shown here for illustrative purposes. 30
4.7 A plot over time of the phase shift after unwrapping of all training data, colored by gesture, showing how there is no general trend of only increasing or decreasing phases. A histogram shows the final phase unwrapped values. The histogram shows the final values are generally positive, but tend to not be at the extremes. 31

4.8 Samples of each gesture from Widar being transformed using each of the three signal-to-image transformations. Each pair of columns is the transformation by a single signal-to-image transformations on the amplitude and phase signals, respectively. 33

4.9 The CNN-based VAE used in DARLInG. The number of ConvBlock elements is variable, depending on the experiment being performed. The reparameterization trick is described in [18]. 35

4.10 Details of the multi-task learning module. The generator is made up of a series of deconvolutional layers and produces the BVP prediction while the classifier is a fully connected network and produces a probability distribution over each gesture class. 37

5.1 The summarized results of the experiments, showing F1-score grouped by the signal-to-image transform used. For each signal-to-image transform, the colors represent the RL algorithm and the encoding used. 46

5.2 The difference in performance of our models on in-domain vs. out-of-domain samples. The in-domain samples were taken from a held-out set of the training samples. The values are grouped by signal-to-image transform and the colors represent the RL algorithm and the encoding used. The shown values indicate the reduction in F1-score of our model on out-of-domain samples w.r.t. the in-domain samples with lower values indicating better performance on the in-domain samples. 47

5.3 This chart shows the performance delta between the null head and the embed head for each experiment. Higher positive values indicate that the embed head achieved a higher F1-score. 48

5.4 This chart shows the overall performance delta between using a probability measure and a one-hot encoding for the domain embedding. The values are grouped by signal-to-image transform and the colors represent the RL algorithm used. The scores used here have also been corrected for their respective algorithm's null head performance, i.e., uses the scores seen in Figure 5.3 instead of in Table 5.4. Higher positive values indicate the embed head achieved a higher F1-score when using the probability measure encoding as opposed to using a one-hot encoding. 49

6.1 Comparison of our method against the Widar 3.0 benchmark and the current SOTA. The space between our work and other works is intentionally placed to reiterate the point that these values are not directly comparable as our data splits, although both representing single-user leave-out, are not comparable. . 52
6.2 t-SNEs of the latent space and domain embeddings produced by PPO with one-hot encoding and MTF transformation. 54
6.3 t-SNEs of the latent space and domain embeddings produced by PPO with PM encoding and MTF transformation. 55
6.4 t-SNEs of the latent space and domain embeddings produced by PPO with one-hot encoding and MTF transformation with extended training time. 56

A.1 Histogram of the length of CSI datapoints. CSI datapoints collected were of different lengths with the majority being less than 2000 samples long, representing samples of around 2 seconds each. 68

B.1 t-SNEs of the latent space and domain embeddings produced by PPO with one-hot encoding and MTF transformation with the initial training time, filtered by User. 74
B.2 t-SNEs of the latent space and domain embeddings produced by PPO with one-hot encoding and MTF transformation with extended training time, filtered by User. 75

List of Tables

5.1 Selection criteria for samples in our experiments with the Widar 3.0 dataset. . . 42

5.2 DARLInG's VAE component's performance on gesture prediction on a single-domain split of the Widar 3.0 dataset. 42

5.3 Hyperparameter tuning results. Each section covers a different grouping of hyperparameters affecting a different component or module of DARLInG. . . . 43

5.4 Final experimental results of the performance of DARLInG by type of RL agent, domain embedding encoding, and signal-to-image transformation. Each pair of columns represents the results of the null head and the results of the embed head in each of the metrics accuracy, precision, and F1-score. The final F1-Score of the evaluation is emphasized in bold type. 45

5.5 Extended training run results of the DDPG agent with MTF transform and one-hot encoding. 50

A.1 Distribution of users vs. room number in the dataset. Note that not all users are recorded in all rooms and that the room distribution is usually unbalanced. Room 3 also has the fewest number of samples by an order of magnitude. . . . 66

A.2 Distribution of users vs. gestures performed. Note that gestures 1-6 are always balanced by user, but some users performed more than others. Gestures 7-9 are performed by only some users and gesture 10 is performed by only users 1 and 2. 67

A.3 Distribution of rooms vs repetitions. All samples were repeated at least 5 times. Specifically, in room 1, some samples were repeated 10 and 20 times. 68

A.4 Distribution of users vs. torso locations. Torso locations are always balanced, by only user 1 performs in torso location 6-8. 69

A.5 Distribution of rooms vs. torso locations. Torso locations are always balanced, but torso locations 6-8 are only performed in room 1. Given that only user 1 performs these locations and the number of samples recorded by user one in Table A.4 is the same as in this table, we can assume only user 1 performed in these locations. 70
A.6 Distribution of Face orientation by room and user. Face orientations are always balanced by both user and room and all face orientations are always performed, although the total number of performances by user may be different. 71
A.7 Distribution of repetitions vs. gestures. No specific gesture was repeated more often, but only gestures 1-6 were repeated 20 times. Gesture 10 specifically also only had 10 repetitions. 72

Chapter 1

Introduction

In this ever more connected world that we find ourselves in, IoT devices everywhere are adding little conveniences to our every day lives. With IoT devices reaching mass adoption with a forecasted 27 billion devices in use by 2025 [14], the dream of ubiquitous computing and sensing is transitioning from a mere dream to the reality of our every day lives. Approximately 19% of new electronic devices bought in 2020 also utilize some form of Wi-Fi radio for communications with a forecasted increase to 24% by 2025 [14].

It is clear that this trend towards integrating computing technology into everyday appliances will only accelerate in the future. The increased convenience and efficiency may be the biggest boon of such technologies. For example, smart thermostats can predict heating requirements and adjust accordingly, leading to lower heating costs in a house while maintaining the convenience of having a well heated space.

With all these connected devices capable of edge computing comes ubiquitous sensing, enabling new modalities of interaction and improving data collection and analytics. It is

Figure 1.1: A WiZ A60 Tunable White light bulb, one example of an IoT smart device with an integrated Wi-Fi Radio [33].

now possible to envision households with complete presence detection coverage, for example, through the use of smart motion sensors, enabling increased efficiency by intelligently identifying which rooms require heating and lighting and which do not.

Always-on voice control systems, such as Amazon Alexa and Google Assistant speakers are also increasingly common, making a connected AI-assistant only one activation word away. The always-connected nature of these sensors also enable the gathering and analysis of vast amounts of user data, potentially providing valuable insights into various aspects of our lives.

One challenge that has continued to plague ubiquitous devices is the lack of a ubiquitous user interface which do not require any input devices. It is not too common to use the end device itself as the input. For example, in the case of smart lights or smart thermostats, the end device would be the light bulbs and heating system, respectively. In both cases, a separate control unit, the light switch or thermostat, respectively, is still required. Even in the case of voice control systems, a dedicated smart speaker is still required and must be placed in every room from which interaction is desired.

With these challenges in mind, Wi-Fi-based sensing provides one potential solution. Many IoT devices already contain some sort of Wi-Fi radio, such as the WiZ smart light bulb in Figure 1.1, and it would be a rather safe assumption that spaces with IoT devices would also have some sort of Wi-Fi infrastructure in place as well. With this in mind, the idea of a gesture-based interface based around Wi-Fi becomes rather appealing. The use of Channel-State Information (CSI) also enables the extraction of finer-grain signals from consumer-grade Wi-Fi radios, enough to enable reliable gesture recognition [1]. This method does, though, suffer from the domain-shift problem, achieving the best accuracy only in cases with a prediction model fine-tuned to a specific person and environment it is used in.

With this thesis, we aim to explore the use of CNN-based Variational Autoencoder (VAE) architectures and domain-shift mitigation methods to improve the state-of-the-art in Wi-Fi CSI-based gesture classification. Specifically, we will look at using preprocessing methods to transform the input signal from CSI into an image using signal-to-image transformations, the use of traditional signal processing algorithms to process the incoming CSI signal, and the use of Reinforcement Learning (RL) to perform domain auto-labeling and provide the VAE decoder with additional information.

1.1 Context and Background

IoT devices are without a doubt increasingly prevalent in everyday life. The Atlas building at the Technical University of Eindhoven (TU/e), for example, uses centrally networked lights

for all of its lighting fixtures powered through Power over Ethernet (PoE). This is at least partially credited as one reason the building has the best efficiency of any academic building in Europe when it was constructed [10].

All over the building, presence detection, in the form of motion detection sensors, is also used to automatically set appropriate lighting levels for each room. This is just one example of how ubiquitous computing and sensing has now entered the mainstream and is no longer a dream of a few enthusiasts and IoT evangelists. Developments in AI and big data processing has also made the usefulness of ubiquitous computing much more evident, legitimizing its use in everyday objects.

With these advances, the question has shifted towards what sort of interface we should utilize to provide an always-available non-intrusive experience for the users. One possible solution is gesture-based interfaces. With any ubiquitous computing and sensing product, especially in the consumer space, minimal setup on the user's part is desired; otherwise the product will not become something which is widely accepted and used.

Wi-Fi gesture recognition can solve these issues, providing a gesture-based interface requiring potentially zero additional setup requirements. As a bonus, this would also be a low-cost solution which many IoT devices already having the necessary hardware for regardless.

There also exists a task group for Wireless Local Area Network (WLAN) sensing called 802.11bf within the IEEE 802.11 working group, the group which sets the standards for WLAN, with members from large companies including Huawei, Qualcomm, and Meta [8]. This shows there is genuine interest in the industry to utilize WLAN for these purposes. With an approval date set for September 2024, it is clear that WLAN sensing is not just some theoretical possibility confined to a lab, but rather a very real technology that may soon become widespread.

To make such an approach possible, we utilize machine learning (ML) to process the incoming CSI data and classify user gestures. Wi-Fi technology, while being very different to radar technology, are both forms of radio technology. Although topologically very different, it naturally, or not so naturally, follows then: can Wi-Fi be used for remote sensing? The answer to this question, according to [1] and [7], is a resounding yes!

There are even commercial products which have become available on the consumer market during the writing of this very thesis that incorporates aspects of this technology. Signify's line of WiZ smart lights uses *SpaceSense* technology, which is a proprietary algorithm that detects motion through "disturbances [in the Wi-Fi signal] created by a person's presence" and is available through a software update [32]. It should be noted, though, that this is only capable of presence detection and not of gesture recognition.

ML approaches, however, suffers from degraded performance when faced with domain-

shifts. When dealing with new, unseen users and environments, gesture classification accuracy degrades significantly [7, 1, 26, 2, 15, 17, 46, 16, 21]. As such, factors to mitigate this domain-shift problem are required in any implementation outside of a pristine laboratory setting.

For the purposes of our thesis, the Wi-Fi information we utilize for gesture classification is known as the Channel State Information (CSI). This comes in the form of a set of complex numbers denoting signal differences for each receiver access point (AP) from each transmitting AP. This complex number can then be represented as a set of amplitudes, or radii, and phase shifts, or angles, when considered as polar coordinates on the complex plane. CSI itself is a description of the multipath effects of a signal traveling from the transmitting AP to the receiving AP. In the realm of WLAN, the estimated CSI of incoming signals is used to correct for these multipath effects, making it possible for the system to adapt to current environmental conditions. Importantly, human activity in an environment also affects the CSI, making it possible to infer activity through CSI.

1.2 Motivation

There are significant potential commercial and practical benefits to enabling domain-agnostic Wi-Fi gesture classification.

The upcoming release of the 802.11bf standard, which standardizes the requisite hardware technology for Wi-Fi sensing, marks a pivotal moment from a commercial standpoint. To fully exploit these emerging technologies, it is of utmost importance to investigate their practical applications.

Wi-Fi sensing is also unique in that the hardware for it is already commercially available and widespread, though not necessarily standardized. As such, it is in a unique place where it is an almost completely software-based solution for preexisting, mass-adopted hardware enabling completely new modes of interaction. This makes it unique and, ultimately, much more commercially desirable as a technology to be adopted.

With respect to domain-agnosticism, the motivating factor is that the largest hurdle for Wi-Fi gesture classification is its loss of performance in new, unseen domains. A model that cannot adapt to such issues will inevitably be nonviable for commercial adaptation lest the end-user be required to perform the same gesture hundreds of times in various positions after installing every single smart IoT device.

We also wish to further the state-of-the-art in domain-agnostic models. The results of this thesis is not only applicable to the field of Wi-Fi gesture classification. A successful domain-shift mitigation technique can be used in other ML tasks where domain-independence is

beneficial. The conclusions of this thesis will hopefully be able to advance the field and provide new insight into what future avenues of research may end up being fruitful.

Finally, a Sci-Fi future where all your devices are controlled through your thoughts might seem like a dream, but in reality, we are not too far from this future. The use of Wi-Fi sensing to detect gestures is one step closer to this futuristic dream and in addition to all the potential commercial and practical benefits this technology could bring, the sheer "coolness" of the technology should not be underestimated as a motivation to perform research.

1.3 Problem Statement

Models already exist to classify gestures through Wi-Fi CSI data. Ideally models are solely influenced by those factors which contribute to the correct classification of the gesture, this is, in reality, not the case. These models are susceptible to the influence of "domain factors", encompassing variables such as the identity of the gesturing subject and the surrounding environment in which the gesture is performed.

These domain factors cause feature domain shifts between the training data and the data encountered during actual use or inference despite both domains containing the phenomenon of interest; for example the gesture we wish to classify. Due to these shifts, performance of the model is degraded [7, 1, 26, 2, 15, 17, 46, 16, 21]. It is thus interesting to be able to create a domain-independent model whose output is independent of the aforementioned domain factors.

There exist proposals to mitigate this degradation including using very large datasets. This is one method especially championed by large companies such as Tesla and OpenAI which have vast resources at their disposal. However, the gathering of such large datasets is only feasible for specific scenarios. In the case of Tesla, for example, having a large fleet of vehicles capable of recording what is essentially supervised training data makes it possible to collect the vast amounts of data required to build a dataset usable for the training of self-driving vehicles [36]. Various OpenAI projects, on the other hand, simply scrape large amounts of websites and collect these as part of their dataset [6]. Within the scope of our research, such extensive data collection methods are non-viable, and we must resort to more novel approaches towards domain-independence.

Most alternative methods to simply using very large datasets rely on a ground truth domain label being provided [17, 43]. In some of these approaches, a "discriminator" network is used to predict domain labels from the latent embedding of the data and an adversarial training procedure is then used. The "generator" which produces these latent embeddings must thus generate embeddings which contain no domain information while still maintain-

ing enough information that a classifier model can use its output to correctly classify target features.

In [21], a method is presented which does not require manually labeled domain labels to be provided. Instead, a CNN encoder and state machine neural network are used and their output is fed into a recurrent neural network (RNN) to provide a classification. The RNN is trained through reinforcement learning (RL) to predict features correctly and independently of domain factors.

We hypothesize that we can extend the RL component of [21] to facilitate domain auto-labeling and eliminate the use of a state machine neural network by using signal-to-image preprocessing methods. Towards these goals, we specify our research questions as follows:

1. To what extent can a reinforcement learning agent utilize the latent signal representation produced by the VAE to accurately produce a latent representation of the domain factors (the *domain embedding*), measured by performance metric difference in a domain factor leave-out cross validation setting between a classifier provided the domain embedding and one which has not?

2. To what extent would changing the domain embedding from a one-hot encoding to a probability measure affect the performance of the classifier, measured by the performance metric difference in a domain factor leave-out cross validation between both domain embedding encodings?

3. To what extent does changing the signal-to-image transformation affect model performance, measured by comparing the performance metric difference in a domain factor leave-out cross validation setting between the different signal-to-image transformations?

The rest of this thesis will present a literature review, background on required knowledge, the proposed methodology, experimental results, and discussions and conclusions of those results.

Chapter 2

Literature Review

In this chapter we review important works in the literature which form the foundation of this thesis. We first discuss the initial set of works which cover Wi-Fi activity detection as well as other related works which do not use CSI data specificially before discussing those works which do utilize CSI data and publicly available datasets for this purpose. We then look at various signal-to-image transformation methods which may be applied to time-series signal data, enabling the use of techniques from the image processing domain. Finally, we look into domain shift mitigation methods and specifically reinforcement learning for domain shift mitigation.

2.1 Wi-Fi for Activity Detection

Past works have investigated the use of Wi-Fi signals generally for the purposes of activity detection.

The first work regarding the use of Wi-Fi signals for the detection of humans subjects we could find is the work of Chetty, Smith, and Woodbridge in 2012 [7]. Their work utilized passive Wi-Fi signals propagating through a building with receivers placed outside the building for presence detection. This method achieved reasonable results and proved that Wi-Fi signals could be used to detect human presence in buildings, although it required the indoor and outdoor APs to be synchronized through wires and was unable to detect precise activities of the human subjects.

The first work we could find discussing the use of Wi-Fi for activity detection is the work from Fadel Adib and Dina Katabi, published in 2013 [1]. This work shows the potential of using signals which could be produced by Wi-Fi APs to detect human activity from through a wall. The most important idea in this work is the elimination of the radio “flash” which

comes with the signal hitting a wall and bouncing back towards the transceiver. Their work focused more on the radar technology implications and not on the use of consumer Wi-Fi APs for gesture detection. They did, though, show that using matched filters was enough to perform rudimentary gesture recognition, given coarse enough gestures.

In the same year, a different group published a paper showing how to use signals in the 2.4 GHz range, i.e., compatible with Wi-Fi transceivers, for simple gesture detection using Doppler shift analysis [26]. This paper proposes the use of a narrowband pulse with a very narrow bandwidth of only a few Hertz and detecting the Doppler shift from the returned signal. Using this method, the researchers were able to identify 9 different gestures with a claimed 94% accuracy.

The same group as [1] also published a separate paper in 2014 detailing the use of a custom-built Wi-Fi based device which could detect course body motions by leveraging the geometric position of its antennas and measuring values through a Time of Flight (ToF) approach [2].

Finally, it is also important to note that IEEE has a task group 802.11bf assigned specifically to standardize Wi-Fi sensing technologies [8]. This group is focused on standardizing the hardware requirements, specifically enabling CSI accessibility and specific measurement procedures, which future devices could implement. Their target is to standardize these requirements for future devices both in the sub-7 GHz range and in the 60 GHz range. They additionally provide suggestions for what methods can be then be used to interpret the data provided, including the use of Fast Fourier Transform (FFT) algorithms to calculate a Channel Impulse Response from CSI and a Doppler FFT, which may be directly used for gesture recognition. The standards for 802.11bf is set to be ratified and published by September 2024.

2.2 Wi-Fi CSI for Gesture Recognition

A selection of works which utilize Wi-Fi CSI data specifically for the task of gesture recognition are discussed in this section.

To the best of our knowledge, the first work discussing the use of CSI for gesture recognition was published in 2015 by He et al. [15]. This work looks into the use of CSI and outlier detection to detect gestures, achieving 92% gesture recognition accuracy on four gestures in a line-of-sight experiment and 88% accuracy in a non-line-of-sight experiment.

The 2019 work titled Person-in-WiFi from Wang et al. proposes the use of an array of three transmitter and three receiver antennas to directly predict body segmentation and pose estimation of persons located in between the aforementioned antennas [39]. In this

work, they used an RGB camera to provide ground truth annotations. The ground truth body segmentation masks were generated using Mask-RCNN while the Body-25 model of OpenPose was used for pose detection. This work shows that body segmentation and pose estimation is possible with only CSI data, achieving an mAP of 0.38 for body segmentation and around 0.1 meter error for join estimation. Qualitatively, the results are quite impressive and it is clear that at the very least, the model performs well given that its input data is one-dimensional.

Using the same dataset, Geng, Huang, and De La Torre published DensePose in 2022 performs similarly, but instead produces UV coordinates of the subjects. This work also provides some interesting preprocessing steps on the raw CSI data to improve prediction performance.

WiGAN, published in 2020, proposes the use of a Generative Adversarial Network (GAN) to augment training data using the generator as well as using the discriminator as a feature fusion and extraction module [16]. The output of the discriminator is then used fed into a Support Vector Machine (SVM) which classifies then gesture seen in the input data. The authors claim that by using the discriminator, which fuses together multiple layers of a CNN module through yet another convolution, their method is able to "learn and recognize the importance of different depth features by itself". On publicly available datasets, WiGAN indeed achieves better results than the competing methods at the time, achieving 98% accuracy on Widar 3.0 with in-domain samples and ≈8–15% lower performance on new domains.

The same group, in the same year, published DeepMV which proposes the use of multiple APs and audio sensors, in the form of ultrasound signals, with a domain discriminator and embedding generator [43]. This work is based on the intuition that the fusion of sensor data from multiple APs and audio sensors placed around the room should provide more data. The embedding generator produces a latent representation of the action which is then used by a fully connected neural network to classify the action being performed while the domain discriminator uses the latent representation to predict the domain of the action. A minimax game is played between the embedding generator and domain discriminator to minimize the maximum accuracy of the domain discriminator while maximizing the minimum accuracy of the action classifier. On their self-collected dataset, they were able to achieve 83.7% mean accuracy, outperforming all other benchmarked methods on this dataset.

The 2021 paper by Zhuravchak et al. proposes the use of an LSTM as a classifier [47]. In this approach, CSI data is provided as an input with a fixed length and the LSTM provides a single output representing the action detected within the input window. Their method

achieves 87.8% accuracy on a self-collected dataset.

Ma et al., published in 2021, proposes the use of a CNN and neural network state machine encoder and a LSTM trained using RL to eliminate the need of domain specific information [21]. This work also contains a curated benchmark of many previous works in this field. Their proposal is the use of a neural network state machine relies on the assumption that there is a temporal dependency within and across CSI segments. For example, it is unlikely that a person who is currently standing will immediately be sitting within the next CSI segment without a sit-down transition segment in between. Although unlikely, the probability is non-zero, due to incontinuities in the data and mislabeling in the ground-truth labels, and thus a neural network state machine is used. The results show >97% mean accuracy on in-domain test samples with a drop of 14–17% on out-of-domain samples.

Additionally, a few bachelors thesis' from the past years at the IRIS research cluster at TU/e have focused on this problem as well. The 2022 thesis by van den Biggelaar proposes the use of reinforcement learning with Deep Q-Networks as the gesture classifier [5]. Their result shows ∼88% mean accuracy on Widar 3.0, dropping by ∼4% on out-of-domain test samples.

The 2022 thesis by Oerlemans compares how different preprocessing methods appear to affect gesture recognition performance [25]. Specifically, signal filtering through a finite impulse response filter and phase unwrapping, transformation to a Doppler frequency spectrum (DFS), and transformation using Gramian Angular Fields (GAF) was explored. On the SignFi dataset, they show that each of the above steps do indeed significantly improve model performance with the GAF transformation coupled with signal filtering resulting in the best performance.

2.3 Wi-Fi CSI datasets for Gesture Recognition

Three Wi-Fi CSI datasets for gesture recognition were considered for this thesis and they are each explained in this section.

The Widar 3.0 dataset, forthwith referred to as Widar for brevity, presents a dataset with developed specifically for "cross-domain learning solutions" [46]. The solution provided in this dataset is two-fold: 1) the (relatively) high number of domains that the data was collected with, and 2) the proposal of Body-coordinate Velocity Profile (BVP) with is a theoretically domain-independent representation of the data. More details of BVP is discussed in Section 3.4. Finally, this paper also provides a baseline model to compare against.

SignFi is a dataset of Wi-Fi CSI data specifically for sign language recognition [22]. This dataset contains over 276 sign gestures in a lab and home environment with five different

users. A baseline CNN model is also presented in this paper, capable of achieving a ∼87% mean accuracy over 150 sign gestures.

Person-in-WiFi is the dataset used in the paper by Wang et al. [39]. This dataset was made public and includes both Wi-Fi CSI data and RGB camera data of the activity from a fixed position. This dataset was collected specifically for pose estimation solutions and is not meant specifically for activity recognition, as no activity labels are provided in the dataset.

2.4 Signal-to-Image Transformations

Different preprocessing methods have been investigated to transform raw time-series signal data into images for deep learning. Three state-of-the-art approaches are Gramian Angular Fields (GAF), Markov Transition Fields (MTF) [40], and Recurrent Plots (RP) [9]. A search of the current body of literature did not yield any research into a direct comparison of these techniques on a common dataset. Instead, a previous unpublished work by the author of this thesis for the Seminar course at the TU/e has shown that these GAF and MTF performed amongst the best of the state-of-the-art signal-to-image transformations [28]. A more thorough description of each of these methods are described in Section 3.5.

2.5 Domain Shift Mitigation Methods

There are a number of papers focusing on domain shift mitigation methods. A selection of these papers, specifically those which are highly related to our problem domain, are discussed in this section.

The 2021 paper by Zinys et al. focuses on the use of GANs for domain shift mitigation, called Adversarial Domain Adaptation (ADA) [48]. In this work, the discriminator attempts to predict gesture and domain while the generator produces sample data. The discriminator is provided a loss function based on ground truth data and accurately identifying generated data while the generator produces sample data which is in the same domain and gesture as its provided input. Their results, tested on Widar show significantly better performance than the baseline Widar model on unseen domains.

Zhang et al., in 2022, proposes the use of federated learning for domain shift mitigation [44]. The concept is to allow for each user to train their own neural networks and using matched average federated learning to combine all user models together. Tested on the Widar dataset, they show performance competitive with state-of-the-art techniques.

Van Berlo et al. discusses attempts at using mini-batch alignment to generate domain

factor independent latent representations of the data [3]. They showed that unfortunately, the proposed mini-batch alignment pipeline did not lead to better performance across domains. The authors believe this may be due to a lack of sufficient domain factor information, leading to poor mini-batch alignment. Alternatively, the assumptions of the underlying probability distributions may be incorrect. In any case, it seems that this method, given current publicly available datasets, does not improve the SOTA.

Finally, the 2022 bachelors thesis by Sips investigates the use of network pruning and quantization for domain shift mitigation [34]. The basic concept is to improve model robustness by enforcing sparsity and utilizing mixed precision training. Tested against a baseline where sparsity and mixed precision training was not used, the modified networks performed slightly worse on in-domain test samples while they had mixed results on out-of-domain samples.

2.6 Domain Shift Mitigation using Reinforcement Learning

The following section contains some selected works in domain shift mitigation, mainly those related directly with RL of self-supervised techniques.

In 2021, Zheng et al. published research into using RL based feature selection for domain shift mitigation [45]. The proposed method would be able to select the most relevant features across two domains by employing Q-learning to learn policies for feature selection, utilizing the performance of a domain discriminator as its reward function. By doing so, they attempt to align the feature manifolds between both domains and produce domain invariant features in the vision field through the use of reinforcement learning. This also has led to our own hypothesis that reinforcement learning can be used for domain auto-labeling. Benchmarked on publicly available datasets, this method achieves the best mean accuracy among SOTA methods.

Martini et al. published a technique called “Domain-Adversarial Neural Networks” in 2021 [23]. This technique uses a feature extractor, a domain classifier, and a label predictor. The feature extractor maps the input to a latent representation, the domain classifier predicts the domain given the latent representation, and the label predictor provides a class prediction given the extracted features. The loss function of the model balances the loss of the label predictor and the domain classifier, both using cross entropy Loss, with the goal of providing a latent representation from the feature extractor which is domain-invariant, yet still discriminative such that the domain classifier is still capable of accurately classifying the domain. Additionally, the paper proposes the use of Maximum Mean Discrepancy (MMD) to measure the distance between domains. The MMD measures the kernel-based distance be-

tween feature means of each domain. This work proposes balancing minimizing the MMD while maintaining the distinctness of each domain, measured by the performance of the domain classifier.

Chapter 3

Background Knowledge

In this chapter, we will explore some of the background knowledge required to understand the methodology chosen and experiments undertaken throughout this thesis. We will first explore Variable Autoencoders (VAEs) before delving into some background about Reinforcement Learning and a discussion of Wi-Fi Channel State Information (CSI) and Body-coordinate Velocity Profiles (BVP). Finally, we will discuss the three signal-to-image transformations that are explored in this work.

3.1 Variational Autoencoders

VAEs, a type of generative model which combines elements of both autoencoders and probabilistic modeling, are suited for unsupervised representation learning and was first introduced by Kingma and Weling in [18]. Standard autoencoders are a network architecture in which data is encoded into a lower-dimensional latent space by an encoder. A decoder then reconstructs the original input from the latent representation. VAEs introduce probabilistic modeling to autoencoders by using a probability distribution over the latent space instead of using the latent space directly.

Its original design aimed at using a generative model as an implicit form of regularization. By forcing the model to learn a representation which is also useful for data generation, the representation learned must have some sort of statistically independent but meaningful representation of the variations in the input data, leading to better performance at both the auxiliary task of data generation as well as the main task of discrimination [19].

Generally, VAEs are described as two coupled but independent models: the encoder and decoder. The encoder is a Bayesian network of the form $q(\boldsymbol{z}|\boldsymbol{x})$ where $\boldsymbol{z}|\boldsymbol{x}$ may be a (deep) neural network. Similarly, the decoder is also a Bayesian network of the form $p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$

where $\boldsymbol{x}|\boldsymbol{z}$ may also be a (deep) neural network. The input signal $\boldsymbol{x}$ is thus represented by $\boldsymbol{z}$.

The purpose of the encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ with parameters ϕ is to produce an approximation of the true, but intractable, posterior. The encoder neural network is then used to produce the set of parameters for latent variables such that

$$(\boldsymbol{\mu}, \log \boldsymbol{\sigma}) = \text{EncoderNeuralNet}_\phi(\boldsymbol{x}) \tag{3.1}$$

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma})) \tag{3.2}$$

where $\mathcal{N}$ is the normal distribution.

The purpose of the decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is to produce a mapping between the latent space $p_\theta(\boldsymbol{z})$ and the original, observed distribution through learning a joint distribution $p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{z})$.

The VAE is optimized through the evidence lower bound (ELBO), incorporating Kullback-Leibler divergence, the details of which are thoroughly explained in [19].

By using the reparameterization trick introduced in [18], the ELBO can then be differentiated with respect to both parameters ϕ and θ at the same time through stochastic gradient descent.

VAEs are used in various areas including image generation, data compression, denoising, and image recognition. By using a latent representation of the data which is statistically meaningful, the generated data is much more likely to come from the same underlying distribution as the original data.

3.2 Reinforcement Learning

Reinforcement learning (RL) is a machine learning paradigm which teaches agents how to make decision in complex environments [35]. The general paradigm is that of an agent which takes actions in an environment as a response to its observations of the current environment state. The environment then provides a reward for the agent and transitions into a new state as a response to the action. The agent is given the goal of maximizing the cumulative reward over time. The main use cases of RL are in robotics, gaming, finance, and healthcare where performing complex tasks or optimizing control systems is necessary.

Throughout this work, we focus on model-free RL. Model-free RL is an approach to RL which focuses on learning directly from interactions with the environment without necessarily modeling the dynamics of the environment explicitly. With this approach, the agent learns policies or value functions which it uses to make decisions. Some well-known model-free RL algorithms include Q-Learning, State-Action-Reward-State-Action (SARSA), Deep Q-

Networks (DQN), Proximal Policy Optimization (PPO), and Asynchronous Advantage Actor-Critic (A3C).

Furthermore, there are two main learning paradigms to RL, namely on- and off-policy learning.

On-policy RL learns directly from data collected by the current policy. This entails updating the policy directly using data, i.e., the reward gathered, of the same policy. This makes it especially suitable for scenarios where data collection is cheap, for whatever measure of cheap is appropriate in the given scenario. By updating the policy directly, the learning trajectory may be more stable due to consistency. On the other hand, this same consistency can lead to reduced exploration of the action space as well as reduced sample efficiency due to the slow exploration. An example of this learning paradigm is PPO, which is one of the two methods used in this thesis.

Off-policy RL, conversely, learns from a *replay buffer* of past experiences. A replay buffer is a store of past experiences that an agent has experienced. During each step taken where the agent interacts with its environment, the action and resulting observation and reward is stored. During the update stage, a random sampling from the replay buffer is then taken and used to update the agent's policy or value function. By doing so, this enables *off-policy* learning, where the agent learns from previous experiences, including those performed under a different policy. This also allows for updates to be performed in batches, increasing throughput. The increased data diversity and ability to use samples from previous policies leads to better exploration and better sample efficiency. One example of off-policy RL is Deep Deterministic Policy Gradient (DDPG).

3.2.1 Proximal Policy Optimization

PPO is a model-free on-policy RL algorithm proposed in Schulman et al. [30]. It is a further development based on Trust Region Policy Optimization [31] and works by iteratively sampling data through interactions with the environment and optimizing a "surrogate" objective function using stochastic gradient ascent. The paper explores two approaches to this "surrogate" function, one using a penalty on KL divergence and one using a clipped objective function.

In the KL divergence approach, for each update of a given policy π_θ, the KL-penalized objective L^{KLPEN} is optimized with

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t - \beta KL\left(\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)\right) \right] \tag{3.3}$$

where t is the current timestep, a the action, s the environment state, $\hat{A}_t$ an estimator of the

advantage function at timestep t, and $\hat{\mathbb{E}}_t$ the empirical average over a finite batch of samples. With every policy update, we also update β by first computing $d = \hat{\mathbb{E}}_t\left(\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)\right)$. If $d < d_{targ}/1.5$, then we half β. If $d > d_{targ} \times 1.5$, then we double β. β is a hyperparemter that can be chosen, but is "not important in practice because the algorithm quickly adjusts it" [30].

In the clipped surrogate objective approach, for each update of the policy, the clipped surrogate objective L^{CLIP} for a given policy π_θ is updated with

$$L^{CLIP}(\theta) = \min\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t, g(\epsilon, \hat{A}_t)\right] \tag{3.4}$$

$$\text{where } g(\epsilon, \hat{A}_t) = \begin{cases} (1+\epsilon)\cdot\hat{A}_t & \hat{A}_t \geq 0 \\ (1-\epsilon)\cdot\hat{A}_t & \hat{A}_t < 0 \end{cases} \tag{3.5}$$

The clipped surrogate object approach shows better empirical performance in [30] and is the one that we use in this thesis.

The algorithm itself is relatively simple and is best understood by directly reading the pseudocode seen in Algorithm 1.

Algorithm 1: Proximal Policy Optimization (PPO) Algorithm

Input: Initialize policy parameters θ_0 and value function parameters ϕ_0

for $k = 0, 1, 2, \ldots$ **do**

- Collect trajectories $D_k = \{\tau_i\}$ by running policy $\pi_k = \pi_{\theta_k}$ in the environment;
- Compute rewards $\hat{R}_t$ for each action;
- Calculate advantage estimates $\hat{A}_t$ using the value function V_{ϕ_k};
- Update the policy by maximizing L^{CLIP}:

$$\theta_{k+1} = \arg\min_\theta \frac{1}{|D_k|T}\sum_{\tau\in D_k}\sum_{t=0}^{T} L^{CLIP} \tag{3.6}$$

  using Adam;
- Fit the value function by regression on mean-squared error over the collected trajectories D_k;

end

3.2.2 Deep Q-Networks

Before we delve into DDPG, we will firsts look at DQNs, which DDPG builds off of and extends to continuous action spaces. DQNs are a model-free off-policy RL alrogithm and are an

extension of Q-Learning, replacing the explicit Q-tables with neural networks [41, 24].

The concept behind DQNs is to use a deep neural network to approximate the optimal action-value $Q(s,a)$ function given by

$$Q^*(s,a) = \max_{\pi} \Big[\mathbb{E}\Big[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t = s, a_t = a, \pi \Big] \Big] \tag{3.7}$$

where r_t is the reward at time t, γ a hyperparameter, $\pi = P(a|s)$ the policy, and s, a the observation and action, respectively. Intuitively, Q^* then is the maximum sum of rewards discounted by γ after making a given observation and taking a given action.

The agent's experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ are stored in the replay buffer $D_t\{e_1, \ldots, e_t\}$. The Q-network is then updated during the learning process in batches of randomly and uniformly sampled experiences from D such that the drawn samples are $(s,a,r,s') \sim U(D)$. A separate target network is introduced as well to address the problem of instability during training. The target network is a copy of the Q-network, but is updated less frequently and often with a slower learning rate.

At each iteration i of the learning process, the loss function which is optimized for is the Bellman equation describing the optimal action-value function and is given by

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim U(D)} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s,a;\theta_i) \right)^2 \right] \tag{3.8}$$

where θ_i are the parameters of the Q-network at iteration i and θ_i^- is the target network parameters. θ_i^- is only updated with the Q-network parameters θ_i every C steps and is otherwise kept fixed.

3.2.3 Deep Deterministic Policy Gradient

DDPG builds on DQNs and extends the action space to continuous action spaces [20]. Extending the action space by simply discretizing a continuous action space is not viable due to the curse of dimensionality that comes with too many output dimensions. DDPG provides a solution by concurrently learning a Q-function, called the critic, as well as a policy, called the actor.

The loss function optimized by the critic in DDPG is modified from DQN to include the actor such that

$$L_Q = \frac{1}{N} \sum_i \Big(y_i - Q_{\theta_Q}(s_i, a_i) \Big)^2, \tag{3.9}$$

$$\text{where } y_i = r_i + \gamma Q'_{\theta_{Q'}}(s_{i+1}, \mu'_{\theta_{\mu'}}(s_{i+1}))N \tag{3.10}$$

with critic network Q and actor network μ with parameters θ_Q, θ_μ, respectively. s_i, a_i, r_i are the observed state, action, and reward at time step i.

The algorithm itself is best described in the pseudocode presented in Algorithm 2.

Algorithm 2: Deep Deterministic Policy Gradient (DDPG) Algorithm

Input: Randomly initialized critic network Q and actor network μ with parameters θ_Q, θ_μ

Initialize target networks Q', μ' with weights $\theta_{Q'} \leftarrow \theta_Q,\ \theta_{\mu'} \leftarrow \theta_\mu$;

Initialize replay buffer R;

for *episode* $k = 0, \ldots, M$ **do**

  Let $\mathcal{N}$ be a normal random process for action exploration Receive initial observation state s_1 **for** $t = 1, \ldots, T$ **do**

    Select action $a_t = \mu_{\theta_\mu}(s_t) + \mathcal{N}_t$ according to the current policy and exploration noise;

    Execute action a_t and observe reward r_t and new state s_{t+1};

    Store transition $e_t = (s_t, a_t, r_t, s_{t+1})$ in R;

    Sample a random minibatch of N transitions $e_i \in R$;

    Set y_i according to Equation 3.10;

    Update critic by minimizing the loss according to Equation 3.9;

    Update the actor policy using the sampled policy gradient

$$\text{let } a_i = \mu_{\theta_\mu}(s_i)$$
$$\nabla_{\theta_\mu} J \approx \frac{1}{N} \sum_i \nabla_{a_i} Q_{\theta_Q}(s_i, a_i) \nabla_{\theta_\mu} a$$

    ;

    Update the target networks:

$$\theta_{Q'} \leftarrow \tau \theta_Q + (1 - \tau)\theta_{Q'}$$
$$\theta_{\mu'} \leftarrow \tau \theta_\mu + (1 - \tau)\theta_{\mu'}$$

    ;

  end

end

3.3 Channel State Information

In the context of Wi-Fi, channel state information (CSI) refers to the current conditions and characteristics of a given wireless communication channel. The current conditions and characteristics are typicially given as a set of complex values $z = re^{i\varphi}$ where, the radius r of the value is interpreted as the amplitude shift of the signal and the angle φ the phase shift of the signal.

This CSI contains data about how the wireless signal has traversed the environment including reflection, diffraction, and scattering as a result of the path the signal has taken. Additionally, information of the multiple paths that the signal has taken, known as its multipath characteristics, is also contained within the signal. Tracking the real-time (or in our case captured temporal) changes in the signal allows us to also capture data about the dynamic behavior of objects and obstacles in the environment the signal has traversed. By taking advantage of this information, we are able to characterize the movement, and by extension gestural data, of subjects which are affecting the signal.

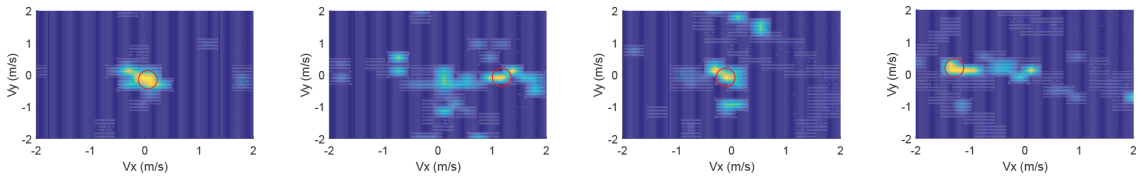
3.4 Body-coordinate Velocity Profile

The CSI information, while useful, is also very heavily subjected to various domain factors including, but not limited to, the body composition of the subject, the various objects and their positioning in a room, the direction the subject is facing, the intensity of solar radiation on a given day, and whether Venus is ascending and in Taurus[1]. To remedy this, Zheng et al. [46] proposes the use of the Body-coordinate Velocity Profile (BVP), a representation of the CSI which is theoretically independent of domain factors [46]. A visual representation of BVP can be seen in Figure 3.1.

BVP is derived through a multi-step process where the CSI is first analyzed to capture both the Doppler Frame Shift (DFS) information and to estimate orientation and location of the subject. The DFS D is then a matrix with dimensions $F \times M$ where F is the number of sampling points in the frequency domain, i.e., the channels, and M the number of transceiver pairs. The full derivation of DFS from CSI can be seen in Equation 2 of [46].

The BVP V is derived from the DFS as a discrete $N \times N$ matrix with N being the number of possible values of velocity components decomposed along each axis of the body coordinates. The authors of [46] use the local body coordinates of the subjects as the origin with the positive x-axis aligning with the orientation of the person. The full details of the BVP

[1] The last two were meant as a joke, but is meant to illustrate how difficult it can be to fully characterize all domain factors which may affect the CSI information and how seemingly small things may have alter the CSI signal entirely



(a) Stage 1: Start (b) Stage 2: Pushing (c) Stage 3: Stop (d) Stage 4: Pulling

Figure 3.1: A series of slices of a BVP representation of a pushing and pulling gesture, sliced along the time axis. The main velocity component, which represents the movement of the subject's hand, is highlighted with a red circle. This set of images is taken directly from [46].

derivation, which are somewhat unnecessary to the understanding of the rest of this thesis, can be found in Section 4 of [46].

Suffice to say, the BVP representation of a gesture is an $N \times N \times T$ tensor, where T is the time dimension. This BVP is thus representative of dynamic effects brought by movement of the subject in the DFS representation of the CSI signal and aligned to the orientation and centered on the location of the subject.

3.5 Signal-to-Image Transformations

In this section, we will look into the mathematical formulation of GAFs, MTFs, and RPs and provide a brief overview of how each of these methods transform a time-series data signal into a tensor which can be interpreted as an image.

3.5.1 Gramian Angular Fields

GAFs [40] represents time series in a polar coordinate system and plots the angles in this system.

First, let X be the input signal of length n and $\tilde{X}$ the input rescaled to the interval $[-1, 1]$.

We then represent $\tilde{X}$ in polar coordinates with arguments for the angle ϕ and radius r as

$$\begin{cases} \phi = \arccos(\tilde{x}_i), & \tilde{x}_i \in \tilde{X} \\ r = \frac{t_i}{N}, & t_i \in \mathbb{N} \end{cases} \tag{3.11}$$

where t_i is the timestamp of a given value of x_i and N a constant factor to regularize the span of the polar coordinates.

This is then transformed into the matrix G of size $n \times n$ as

$$G = \begin{Bmatrix} \cos(\phi_1 + \phi_1) & \cdots & \cos(\phi_1 + \phi_n) \\ \cos(\phi_2 + \phi_1) & \cdots & \cos(\phi_2 + \phi_n) \\ \vdots & \ddots & \vdots \\ \cos(\phi_n + \phi_1) & \cdots & \cos(\phi_n + \phi_n) \end{Bmatrix} \qquad = \tilde{X}' \cdot \tilde{X} - \sqrt{I - \tilde{X}^2}' \cdot \sqrt{I - \tilde{X}^2} \tag{3.12}$$

where I is the unit row vector $[1, 1, \ldots, 1]$. Defining the inner product $< x, y >= x \cdot y - \sqrt{1 - x^2} \cdot \sqrt{1 - y^2}$ results in G being a Gramian matrix with

$$G = \begin{Bmatrix} < \tilde{x}_1, \tilde{x}_1 > & \cdots & < \tilde{x}_1, \tilde{x}_n > \\ < \tilde{x}_2, \tilde{x}_1 > & \cdots & < \tilde{x}_2, \tilde{x}_n > \\ \vdots & \ddots & \vdots \\ < \tilde{x}_n \tilde{x}_1 > & \cdots & < \tilde{x}_n, \tilde{x}_n > \end{Bmatrix} \tag{3.13}$$

The GAF provides a way to preserve temporal dependencies, with time increasing along the diagonal from the top left to the bottom right.

3.5.2 Markov Transition Fields

MTFs [40] encode the dynamical transition statistics of a time series by representing these transitions as Markov transition probabilities. Let the input signal X of length n be quantized into Q bins. Each sample x_i from the input signal is then placed in the corresponding bin $q_j, j \in [1, Q]$. A matrix M of size $n \times n$ is then constructed where M_{ij} denotes the probability of a transition between bins $q_i \rightarrow q_j$ between two sequential time steps. Let the quantile bins that contain the data at timestep i and j be q_k and q_l, respectively. This formally defines M as

$$M = \begin{Bmatrix} w_{kl|x_1 \in q_k, x_1 \in q_l} & \cdots & w_{kl|x_1 \in q_k, x_n \in q_l} \\ w_{kl|x_2 \in q_k, x_1 \in q_l} & \cdots & w_{kl|x_2 \in q_k, x_n \in q_l} \\ \vdots & \ddots & \vdots \\ w_{kl|x_n \in q_k, x_1 \in q_l} & \cdots & w_{kl|x_n \in q_k, x_n \in q_l} \end{Bmatrix} \tag{3.14}$$

M is then an image of size $n \times n$ and can be interpreted as representing the probabilities of any state transitioning into another state over time.

3.5.3 Recurrence Plots

RPs [9] transform the image by representing distances between extracted trajectories in the original time series. Let the input signal X of length n have extracted trajectories

$$\vec{x}_i = \left(x_i, x_{i+\tau}, \ldots, x_{i+(m-1)\tau}\right), \ \forall i \ in \{1, \ldots, n - (m-1)\tau\} \tag{3.15}$$

where m is the dimension of the trajectories and τ is the time delay. We can then construct the recurrence plot R as the a matrix of size $(n-(m-1)\tau)\times(n-(m-1)\tau)$. Every value in this matrix is defined as the pairwise distance between trajectories, formally

$$R_{i,j} = \Theta\left(\epsilon - ||\vec{x}_i - \vec{x}_j||\right),\ \forall i,j \in \{1,\ldots,n-(m-1)\tau\} \tag{3.16}$$

where ϵ is a threshold and Θ is the Heaviside function given by

$$\Theta(x) := \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{3.17}$$

This can be interpreted as representing the pairwise distance between trajectories which are above a certain threshold.

Chapter 4

Methodology

This chapter will first discuss the details of the dataset which we are using. We will discuss our approach to the problem of domain agnostic Wi-Fi CSI gesture classification and delve into the details of our chosen architecture. This will first begin with a general overview of our method, then discuss each component of the architecture individually as well as their motivations and intuitions.

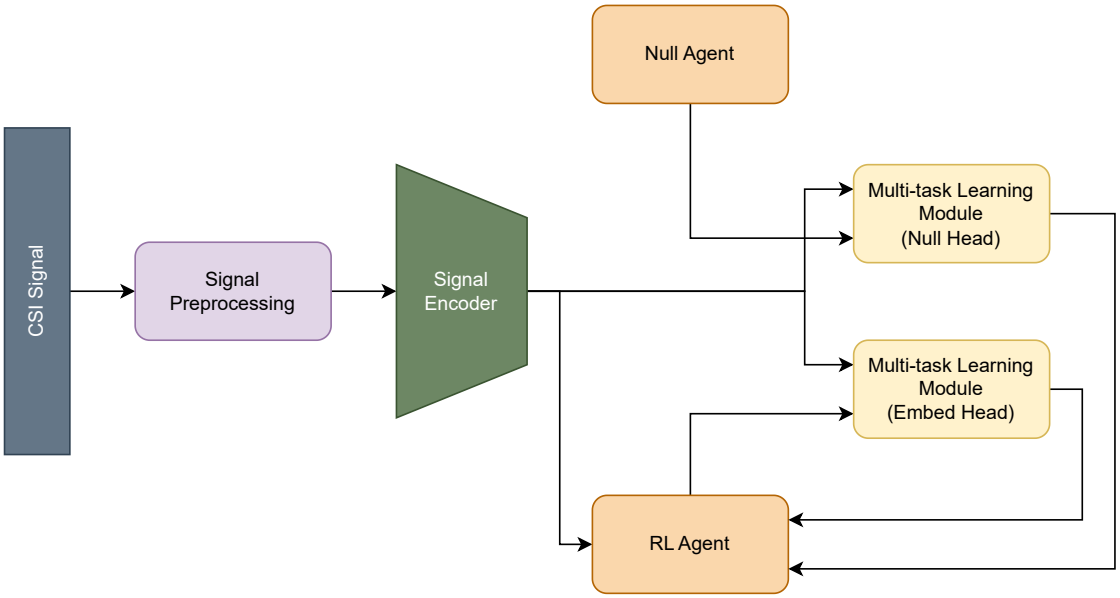


Figure 4.1: An abstracted diagram of the proposed DARLInG architecture.

4.1 Widar 3.0

Widar[46] contains gesture data collected over a series of months in 3 different rooms with 17 different subjects. 6 gestures were performed by every subject. For each gesture performed, the gesture was performed in one of 5 locations with 5 orientations. Further details of the distribution of domain factors can be found in Appendix A. Further details of how the data was gathered can be found in [46].

Each recorded data sample is a time-series consisting of the CSI between a transmitting Wi-Fi access point (Tx) and a receiving Wi-Fi access point (Rx). Each Rx access point (AP) has two antennas, each of which are collecting separate time-series streams. The positional

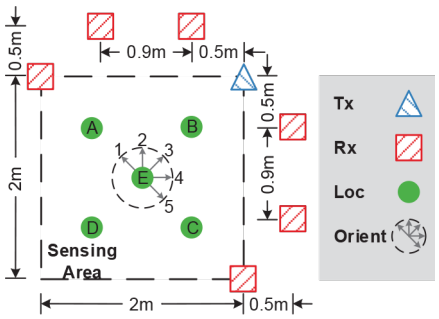


Figure 4.2: Typical positioning of the access points and subjects in Widar. The image is sourced from [46]. Loc shows the different torso locations used and orient the different orientations the subject may be at.

setup of the gesture collection spaces can be seen in Figure 4.2.

The data itself is provided as a collection of both CSI data dumps, collected at 1000 Hz, as well as BVP tensors, collected with a temporal resolution of 10 Hz. Each CSI data dump contains samples of sets of complex values with variable length, the shortest having a length of around 100 samples and the longest with a length of around 3100 samples. As the majority of the data have a length of less than 2000 samples, we set the cutoff to 2000 samples to ensure that every datapoint we use for training has the same length. We pad those of shorter length with 0s and we simply apply a cutoff on those datapoints which are longer.

Due to our focus on looking at single domain factor leave out as our main research question, we have chosen the task of single-user leave out. To ensure that we are focusing only on this domain factor, our selection parameters for the training set and test set are seen in Table 5.1b.

4.2 DARLInG

In this thesis, we present our novel approach DARLInG (Domain Autolabeling through Reinforcement Learning for the Identification of Gestures). DARLInG is our proposed approach to domain-independent gesture identification using RL. Our intuition comes from [45] in which RL was used to identify features in the data which was invariant to domain shifts. Analogously, we hypothesize that RL can then be used to identify those features which are *not* invariant to domain shifts and produce a *domain embedding* from said features. We hypothesize that by providing our gesture discriminator with not only the original signal, encoded by a CNN-based VAE encoder, but also this domain embedding, the gesture discriminator would be able to significantly increase its performance in gesture recognition throughout multiple domains.

The general architecture of DARLInG can be seen in Figure 4.1 and the code can be found publicly on GitHub[1]. We first propose a signal processing and signal-to-image transformation pipeline, described in Section 4.3. This transforms the input signal into an image that is then passed through a CNN encoder, as described in Section 4.4, encoding the signal into a latent space. This latent representation is then used in two different ways: As an input for the multi-task learning heads, described in Section 4.5, and as the state observation for our RL agent, described in Section 4.6.

Recall first that RL is typically modeled as a Markov decision process with an environment and observed state s_t of said environment at time t. An agent then performs an action a_t and

[1] https://github.com/yvan674/DARLInG

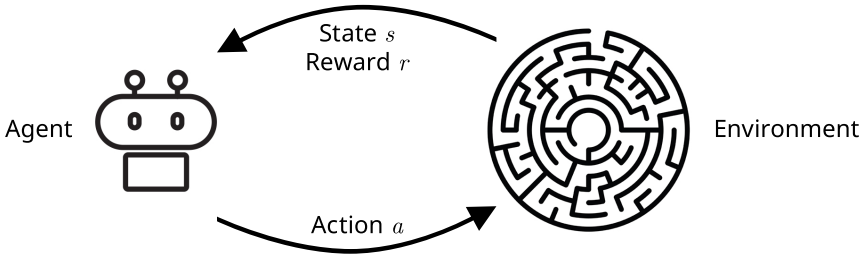


Figure 4.3: The basic paradigm of Reinforcement Learning where an agent interacts with its environment through actions a and receives a new state s and reward r in return.

is provided with a reward r_t and a new observed state s_{t+1}. A typical RL scenario is framed in this way and is visually represented by Figure 4.3.

To mitigate domain-shift, our RL agent is tasked with producing the best possible domain embedding of the domain $\boldsymbol{d}_r = a, \boldsymbol{d}_r \in [0,1]^e$ as its action where e is the dimensionality of the domain embedding. The domain embedding, or action, is generated by the RL agent given the signal latent representation $\boldsymbol{z} = s, \boldsymbol{z} \in [0,1]^f$ as its state observation, where f is the dimensionality of the latent representation.

To provide the RL agent with a reward function we use two different multi-task learning modules. The details of the reward function are described in Subsection 4.6.2. One of these modules is provided $\boldsymbol{d}_\emptyset$, representing a vector of zeros when $\boldsymbol{d}_r$ is one-hot encoded or a value of $\frac{1}{f}$ when $\boldsymbol{d}_r$ is a probability measure. The other module is provided with $\boldsymbol{d}_r = a$.

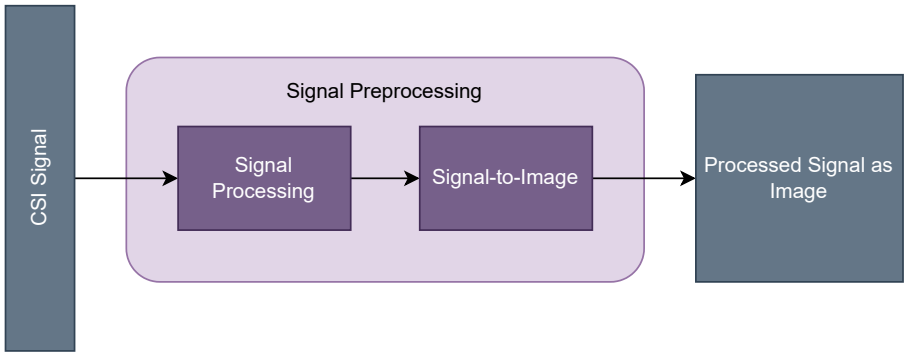
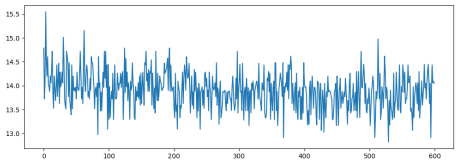
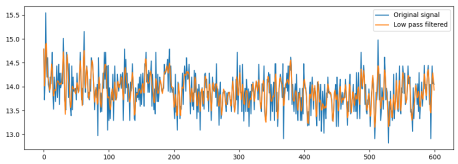


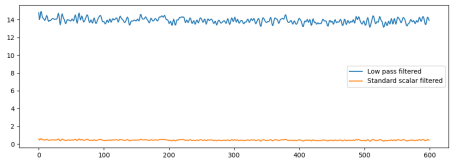
Figure 4.4: Details of the signal preprocessing module. The module is comprised of traditional signal preprocessing and a signal-to-image transformation.



(a) The original amplitude signal.



(b) The original amplitude signal and the low pass filtered signal.



(c) The low pass filtered signal and the signal after the standard scalar.

Figure 4.5: An example of the amplitude shift signal after being transformed by each step of the signal processing pipeline. Only the first 600 samples of one channel from one tranceiver link is shown here for illustrative purposes.

4.3 Signal Preprocessing

As the old adage goes, garbage in, garbage out. As such, we propose a signal preprocessing module, visualized in Figure 4.4. This module first cleans the CSI signal using traditional signal processing and then transforms it into an image for use by the signal encoder.

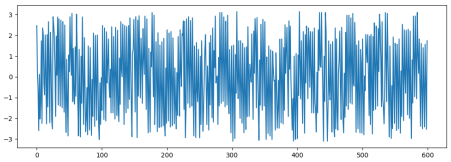
4.3.1 Signal Processing

The CSI data is provided as a set of complex values sampled at 1000 Hz. We first decompose this into its radius and angle components on the complex plane. These correspond to the amplitude and phase shift of the signal, respectively. We process this signal through two different pipelines, one for the amplitude component and one for the phase component, as the phase requires a few additional processing steps compared to the amplitude.

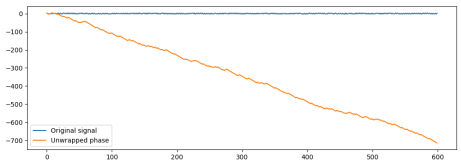
The pipeline for amplitude consists of the following steps:

1. **Low Pass Filter** set to a cutoff frequency of 250 Hz and an order of 4.
2. **Standard Scalar** which has been trained on all training data and transforms the signal to have a mean of 0 and a standard deviation of 1.

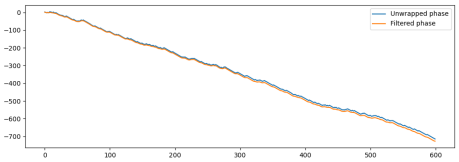
The low pass filter is used to eliminate noise inherent in CSI data from environmental factors. It has been set to these values based on empirical experimentation which shows



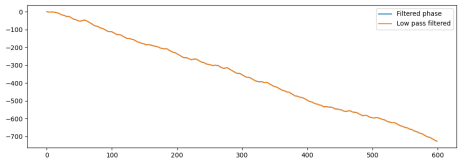
(a) The original phase signal.



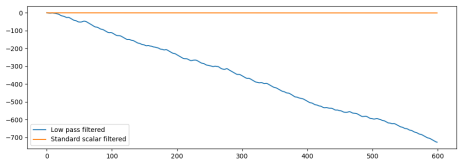
(b) The original phase signal and the phase unwrapped signal.



(c) The phase unwrapped signal and the phase filtered signal.



(d) The phase filtered signal and the low pass filtered signal.



(e) The low pass filtered signal and the signal after the standard scalar.

Figure 4.6: An example of the phase shift signal after being transformed by each step of the signal processing pipeline. Only the first 600 samples of one channel from one tranceiver link is shown here for illustrative purposes.

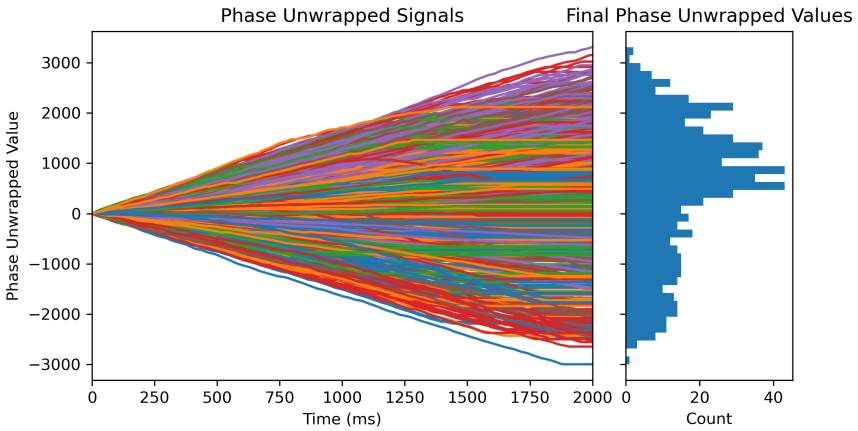


Figure 4.7: A plot over time of the phase shift after unwrapping of all training data, colored by gesture, showing how there is no general trend of only increasing or decreasing phases. A histogram shows the final phase unwrapped values. The histogram shows the final values are generally positive, but tend to not be at the extremes.

that no human movements during the performance of the 6 gestures which we test for has a frequency above 250 Hz.

The standard scalar is used since empirical evidence shows that neural networks work best when input values are close to the interval $[-1, 1]$ [12]. The standard scalar we use is fitted on the training set of the split that is used during a given experiment.

The effect of each step of the amplitude pipeline can be seen in Figure 4.5.

The pipeline for the phase consists of the following steps prepended to the amplitude pipeline:

1. **Phase Unwrapping** which unwraps the phase and makes the signal continuous.
2. **Phase Filtering** step, which applies a uniform and median filter onto the phase shift signal.

We use phase unwrapping to avoid sharp discontinuities in the phase value whenever a phase wraps around from $-\pi$ to π or vice versa. This results in a continuous signal instead of disjointed segments of the signal. We use phase filtering as inspired by [25] with the same parameters and for the same reasons. The effect of each step of the phase pipeline can be seen in Figure 4.6.

The signal can now be considered clean, or at least clean enough that we can continue with further steps.

During the course of experimentation, we considered taking the derivative of the phase, to eliminate a generally monotonously increasing or decreasing signal, as can been seen in

the example in 4.6 after phase unwrapping. Further investigation showed that this is not necessary as most signals do not monotonously increase or decrease. A plot of the phase shifts after unwrapping of all signals can be seen in Figure 4.7. The plot clearly shows that there is no general trend of phases increasing or decreasing infinitely, with many phase shifts staying slightly positive after unwrapping but few signals trending towards the extreme ends of the distribution.

4.3.2 Signal-to-Image Transformation

The next stage in our method is transform the signal into an image, leveraging advances from computer vision. We will experiment with the following three methods for signal-to-image transformation: Gramian Angular Fields (GAF) [40], Markov Transition Fields (MTF) [40], and Recurrent Plots (RP) [9]. A more comprehensive description of each of these transformations can be found in Section 3.5.

We process these images in Python using the pyts package [11] which conveniently contains ready-to-use implementations of all three of the aforementioned transformations. Regardless of the chosen transformation, the signal is transformed into a two-dimensional tensor which can be treated as an image. We also downsample this image to a size of 40×40 pixels for computational complexity purposes, which still provides a temporal resolution $4\times$ that of the provided BVP data in Widar. A visualization of each of the signal-to-image transformations on randomly selected samples of the Widar 3.0 dataset can be seen in Figure 4.8. As can be seen, each CSI sample is transformed into an amplitude as well as a phase image. We can then proceed with the encoding of this image into a latent space.

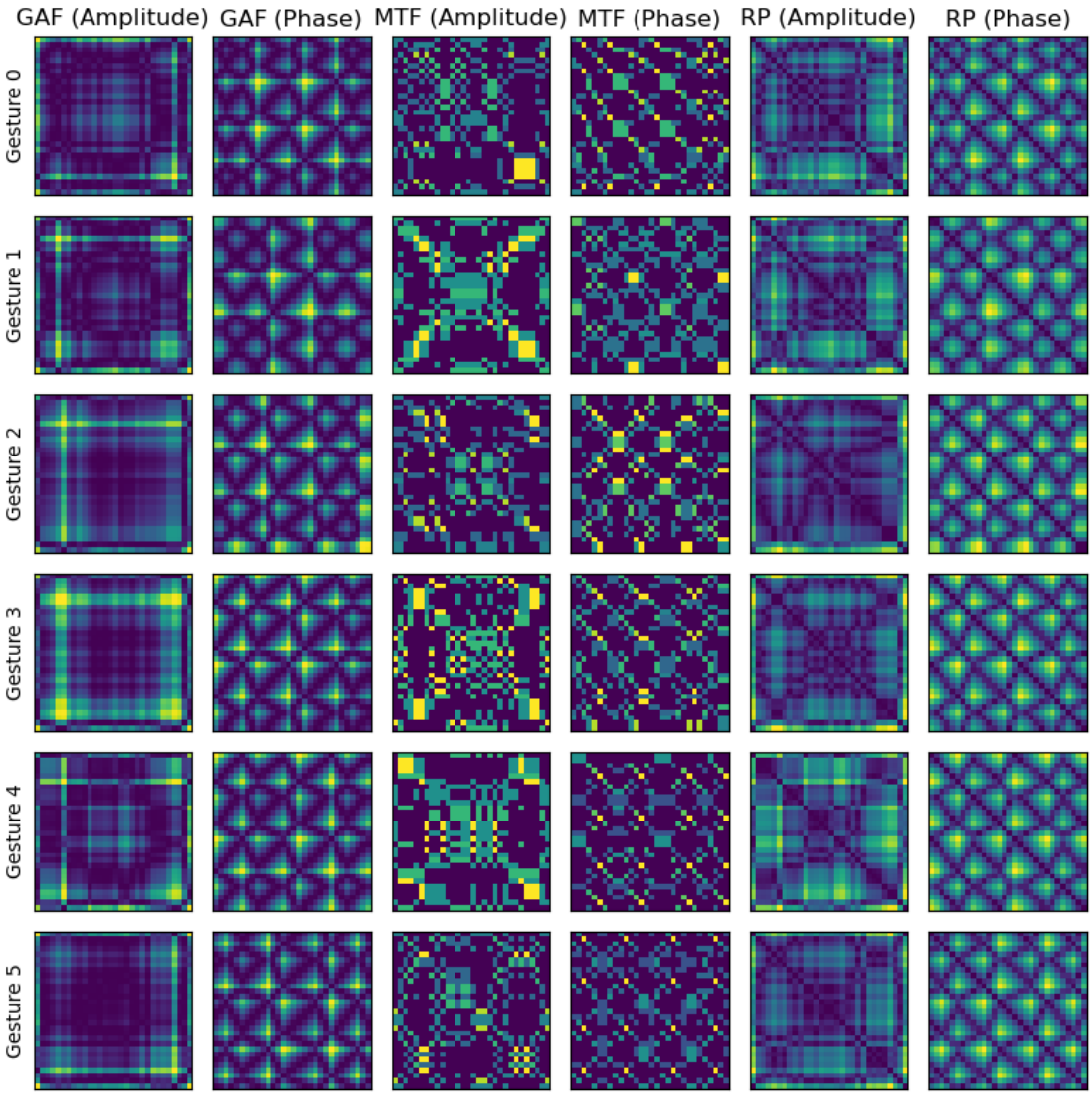


Figure 4.8: Samples of each gesture from Widar being transformed using each of the three signal-to-image transformations. Each pair of columns is the transformation by a single signal-to-image transformations on the amplitude and phase signals, respectively.

4.4 Signal encoding

The encoding of the signal, now an image, into a latent space is performed using a Convolutional Neural Network (CNN)-based VAE. The CNN itself is structured as a standard, deep CNN, made up of multiple `ConvBlocks` where each block is made up of a `Conv2d` with dropout, a batch norm layer, and an activation layer. The number of input channels, filters, and kernel size of the 2-D convolutional layer and its dropout is set dynamically and differs by experiment. The activation function also differs by experiments and is either ReLU, LeakyReLU, or SeLU. The number of `ConvBlocks` is set dynamically and differs by experiment.

These blocks are then followed by two sets of two fully connected layers with a hidden layer size of 8192. One set of these fully connected layers serve to predict the μ and the other set to predict the $\log\sigma$ of the latent variables. The reparameterization trick is then used, adding a random variable $\mathcal{N}(0,1)$ to produce the latent variables $\boldsymbol{z}$. A visualization of the CNN-based VAE used can be seen in Figure 4.9.

The latent representation $\boldsymbol{z}$ of this signal is then passed to both the reinforcement learning agent as its state observation as well as to the multi-task learning modules.

In the case of CSI data, as the signal is provided as both an amplitude and phase image, we have two signal-encoding modules running in parallel; one for the amplitude image and one for the phase image with no sharing of parameters. How the RL agent uses this state observation is described in Section 4.6 while its use by the multi-task learning module is described in Section 4.5.

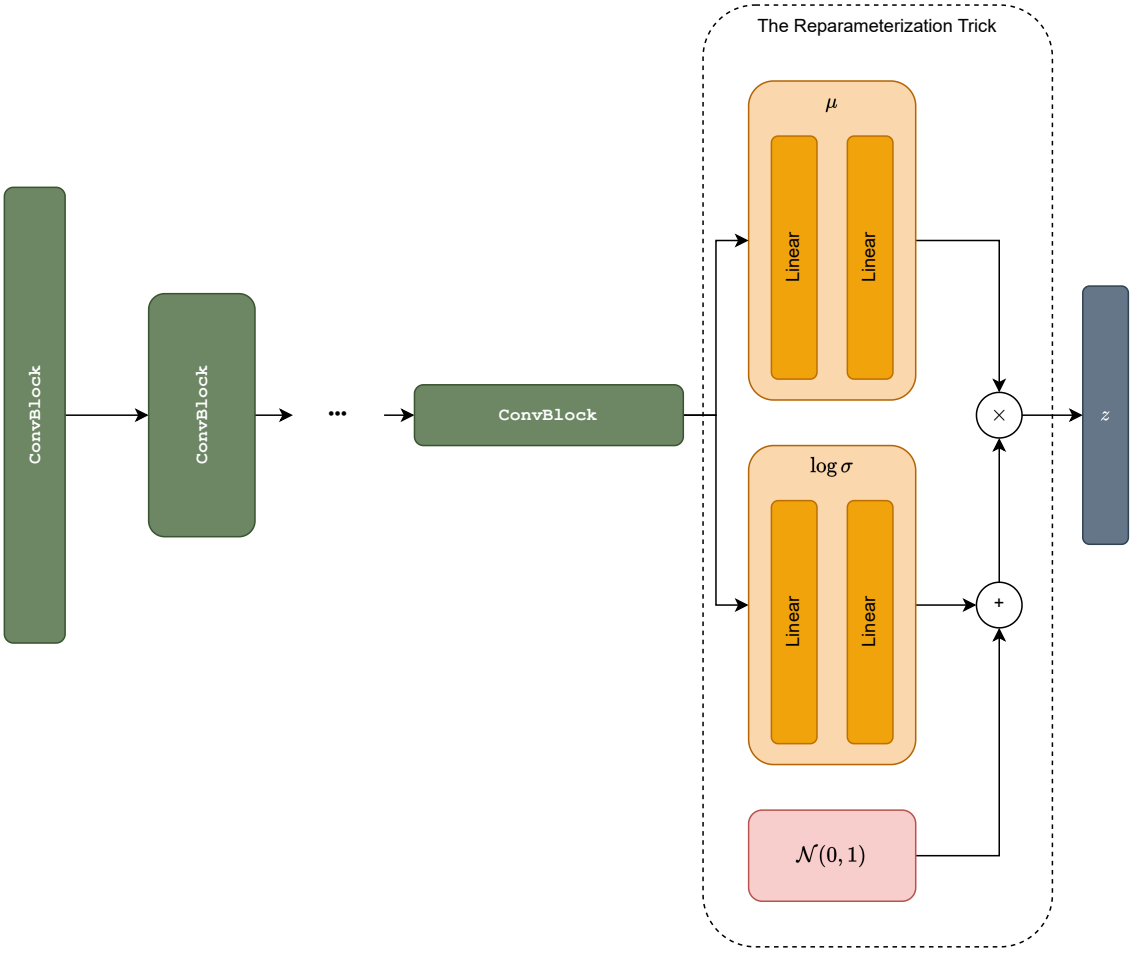


Figure 4.9: The CNN-based VAE used in DARLInG. The number of ConvBlock elements is variable, depending on the experiment being performed. The reparameterization trick is described in [18].

4.5 Multi-task Learning

The actual gesture classification module is built using a multi-task learning module, a visualization of which can be seen in Figure 4.10. The idea, intuitively, is to enforce some sort of representation that is already approaching a domain-invariant representation of the data. To do so, we use the generation of the BVP as an auxiliary task while keeping gesture recognition as our main task. This is done as BVP is theoretically domain-independent [46], even though we are ultimately not interested in the BVP. This approach is inspired by Martini et al. [23], which, although using a different metric to modulate domain-independence as an optimization objective, suggests that an adversarial approach modulating both a domain-independence objective and a domain-discrimination objective can be powerful and increase classification performance. The BVP generation is done by a series of deconvolutional layers while the classifier is a series of fully-connected layers. Additionally, it has been shown empirically that multi-task learning can produce better results in each target task as opposed to dedicated networks [38]. This is likely due to the encoder being guided towards producing a more robust or efficient latent representation capable of being used for many different tasks instead of a latent representation focused on only one task.

Therefore, as BVP generation is a theoretically domain-independent process, we should expect the latent representation to contain both features which are completely domain independent as well as features which are not. We do not, however, want to enforce domain-independence directly on the latent representation, as this may result in worse performance [3].

As seen in Figure 4.1, we utilize two multi-task learning modules. The module receiving $\boldsymbol{d}_{\emptyset}$ is termed the *null head* while the module receiving $\boldsymbol{d}_{r}$ is termed the *embed head*. The motivation behind the use of two heads is such that we can measure the RL agent's performance in an unsupervised manner by comparing the performance of the null head to the embed head. Intuitively, we should expect the embed head to perform better if the embedding $\boldsymbol{d}_{r}$ provided by the RL agent is able to provide some useful information, likely with respect to the domain, as opposed to the null head, which receives no useful information.

For the loss function, we use cross entropy loss on the gesture classifier and mean squared error loss on the BVP generation.

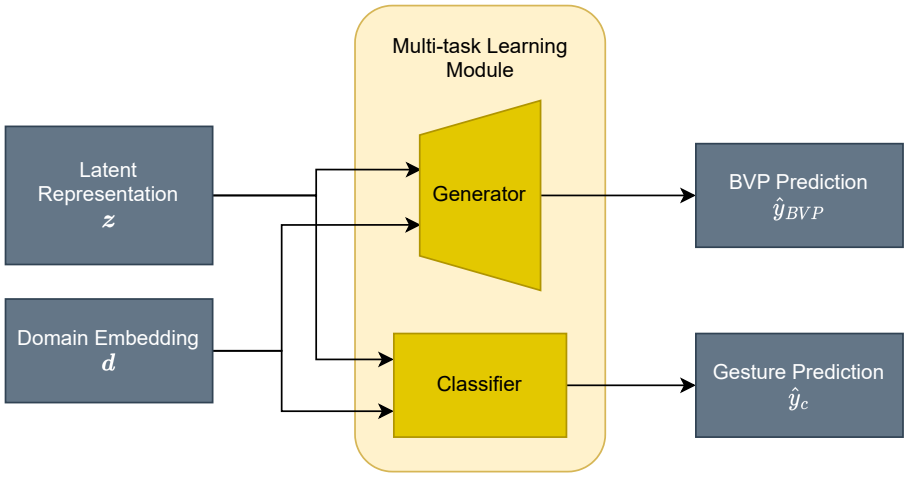


Figure 4.10: Details of the multi-task learning module. The generator is made up of a series of deconvolutional layers and produces the BVP prediction while the classifier is a fully connected network and produces a probability distribution over each gesture class.

4.6 Reinforcement Learning Agent

To mitigate domain shift, we implement a novel method for unsupervised domain autolabeling through reinforcement learning. Recall the common terminology used in RL, the agent, the action a, the state observation s, the reward r, and the learning paradigms discussed in Section 3.2. For our agent, we choose to use the output of the encoder $\boldsymbol{z}$ as its state observation s. The agent then performs an action a which is what we refer to as the domain embedding $\boldsymbol{d}_r$. This domain embedding is either continuous, in the case where the $\boldsymbol{d}_r$ is understood as a probability measure, or discrete, in the case where the $\boldsymbol{d}_r$ is understood as a one-hot encoding of domain factors. The produced $\boldsymbol{d}_r$ is then concatenated to one copy of $\boldsymbol{z}$ and fed to the embed head, while the other head receives a copy of $\boldsymbol{z}$ concatenated with $\boldsymbol{d}_\emptyset$. The reward r is produced as a function of the outputs of both heads, and will be discussed further in Subsection 4.6.2.

In this work, we will be looking at using both PPO and DDPG, on-policy and off-policy algorithms, respectively. Details of PPO can be found in Subsection 3.2.1 while the details of DDPG can be found in Subsection 3.2.3. We use the ready-to-use implementation from Stable Baselines 3 [27], which also requires interpreting the model itself as a Gymnasium environment. Gymnasium is an open-source RL framework previously mainted by OpenAI, but now actively maintained by the Farama Foundation [37].

Our use of RL to tackle this problem is inspired by both [21] and [45]. In [21], as discussed in Section 2.2, RL was successfully used to eliminate the need for domain-specific information Wi-Fi CSI gesture recognition. On the other hand, as discussed in Section 2.6, [45] used

RL for domain-independent feature selection. We therefore were motivated to investigate whether RL could also be used to perform the opposite: to extract relevant domain-specific features and produce a useful embedding of said features for gesture recognition.

4.6.1 Known Domain Encoding

We also used a "known" domain encoding as our baseline. This is a 33-dimensional one-hot encoding of the known domain factors (i.e., twenty users, three room, five face orientations, and five torso locations). For example, in the case of user 2 in room 1 with face orientation 4 and torso location 3, the known domain embedding would be

$$
\begin{aligned}
\boldsymbol{d}_r = [&0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 && \text{user encoding} \\
&1,0,0 && \text{room encoding} \\
&0,0,0,1,0 && \text{orientation encoding} \\
&0,0,1,0,0] && \text{location encoding}
\end{aligned}
$$

This was used as a baseline to compare our RL agents against. The intuition behind this is that our initial hypothesis is essentially "providing domain labels can increase model performance". Therefore, the known domain factors, while incomplete, should provide *some* baseline to which we can compare our model again. We do not need a specific baseline of a model with no domain labels since this is already incorporated into our algorithm in the form of the null head.

4.6.2 Reward Function

As with any RL algorithm, one of the most important components is the reward function used. We experiment with two different reward functions: One we call the *contrastive reward* R_{CON} and one we call *distance-maximization contrastive reward* R_{DMC}.

Contrastive reward Let ϱ denote some metric subject to minimization to measure the gesture classification performance of a multi-task learning module given the gesture classification prediction $\hat{y}_c$. The contrastive reward function R_{CON} is then

$$
R_{CON}(\hat{y}_{c,r}, \hat{y}_{c,\emptyset}, y_c, \gamma) = \gamma \cdot \left(\varrho\left(\hat{y}_{c,\emptyset} - y_c\left(\phi\left(\hat{y}_{c,r}, y_c\right)\right)\right)\right) \tag{4.1}
$$

where y_c is the ground truth gesture classification, $\hat{y}_{c,r}$ the predicted gesture from the embed head, $\hat{y}_{c,\emptyset}$ the predicted gesture from the null head, and γ a multiplier factor chosen during hyperparameter tuning. In practice, we use cross entropy loss as our performance

metric ϱ. As such, our reward function R_{CON} calculates the gesture recognition performance difference between the embed and null heads.

Intuitively, we use this as our metric as the reward is based on the performance difference between the two heads. As the only difference in inputs between the two heads is that one receives $\boldsymbol{d}_r$ while the other receives $\boldsymbol{d}_\emptyset$, we should expect the difference in performance to come solely out of a difference in how powerful the domain embedding provided by the RL agent is in describing the domain factors influencing the CSI signal. Thus, maximizing this difference implies better total model performance. We do not use an absolute value, as we want to penalize the model if the domain embedding results in the embed head performing worse than the null head.

This reward can be thought of as being similar to testing the null hypothesis, where H_0 is provided by the null head and H_1 is provided by the embed head. Our hypothesis, that the domain embedding will provide better results, is measured by the performance difference between then null and embed head, by proxy. If the RL agent provides a useful domain embedding, then the reward will approach $\gamma \cdot \varrho\left(\hat{y}_{c,\emptyset}\right)$.

Distance-Maximization Contrastive Reward Empirical results show that the contrastive reward was not able to produce good performance. After inspecting the embeddings provided by our RL agent, we observed that the difference in value between each dimension was quite minimal, i.e., the agent produced results similar to $\boldsymbol{d}_\emptyset$. We believe this is due to the agent attempting to minimize the penalty for worse performance without being incentivized enough to achieve positive rewards.

In response, we introduce our *distance-maximization contrastive reward* R_{DMC}. With this reward function, we hope to incentivize the agent to produce domain embeddings which are more informative by maximizing the difference between each dimension of the domain embedding. The idea here is to provide stronger signals for the multi-task heads. We do this by first calculating the pairwise difference between each of the dimensions. We then apply a threshold to the difference, where any value below α is zeroed out, similarly to the procedure used in Lasso regression. Finally, we sum together all differences and divide by the number of non-zero values and add the contrastive loss. Formally, this is given by

$$R_{DCM}(\hat{y}_{c,r}, \hat{y}_{c,\emptyset}, y_c, \boldsymbol{d}_r, \gamma) = \gamma \cdot \frac{\sum_{i \le e} \sum_{j \le e, j > i} \left(|d_{r,i} - d_{r,j}|\right) \cdot \mathbb{1}_{>\alpha}\left(|d_{r,i} - d_{r,j}|\right)}{\sum_{i \le e} \sum_{j \le e, j > i} \mathbb{1}_{>\alpha}\left(|d_{r,i} - d_{r,j}|\right)} + R_{CON}(\hat{y}_{c,r}, \hat{y}_{c,\emptyset}, y_c) \quad (4.2)$$

$$\text{where } \mathbb{1}_A(x) := \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases}, \quad (4.3)$$

e the dimensionality of the domain embedding $\boldsymbol{d}_r$, $\mathbb{1}_A(x)$ the characteristic indicator func-

tion, and α a hyperparameter to tune. $d_{r,i}$ can be understood as the magnitude of the i-th dimension of the domain embedding. In practice, we set $\alpha = 0.1$.

4.7 Model Training

The general training process of the model is described in Algorithm 3. We set the training of the RL agent to begin only at epoch ζ to ensure that the latent representation provided by the encoder has somewhat stabilized and actually contains useful information. Otherwise, the RL agent will be training on garbage observation states. This is especially relevant for DDPG, as the replay buffer may contain observations from those garbage environment states even during the later stages of training. We alternate training between the RL agent and the VAE, rather than training both simultaneously, as we believe this will lead to more stable performance.

Algorithm 3: Simplified DARLInG Training Algorithm

Input: Initialize parameters of the encoder q, the decoder of one multi-task learning module p, and the gesture classifier of one multi-task learning module g

Input: ζ, the epoch to begin phase 2 of training

```
for k = 0, 1, ..., N do
    if k ≥ ζ then
        if k = ζ then
            Duplicate the multi-task learning module and its optimizer. The original
             multi-task learning module is designated the null head and the new
             module the embed head;
            Initialize the RL agent;
        Freeze the parameters of the encoder q and both the null and embed head;
        Train the RL agent for h epochs with samples from the training dataset, using
         the latent representation of a datapoint encoded by the encoder as a state
         observation;
        Freeze the parameters of the RL agent;
        Unfreeze the parameters of the encoder and both the null and embed heads;
    Train the encoder and all multi-task learning modules on the training set;
```

Chapter 5

Experiments

To investigate our research questions, we ran a series of experiments. Recall the three research questions we wish to answer from Section 1.3, which can be summarized as: 1) How well does DARLInG perform overall, 2) what is the performance difference in using a one-hot encoding compared to a probability measure (PM) representation of the domain embedding, and 3) how does changing the signal-to-image transformation affect model performance?

In this chapter, we will first take a look at the preliminary experiments performed to verify that our model is adequate for the task at hand. We will then discuss the experimental setup used to answer our research questions as well as the results of the experiments.

5.1 Preliminary Experiments

We perform two preliminary experiments: running the VAE component only on a single-domain split of the dataset to ensure that the model is actually capable of classifying gestures and hyperparameter optimization to ensure that our experiments are being run with optimal hyperparameters.

5.1.1 Single-Domain Experiments

To ensure that the model is performing adequately for the purposes of this experiment, we first chose to run the VAE component with no RL component on a single-domain split of the data. This is done to ensure that the model can actually perform gesture classification in the first place, given no domain shifts.

In this experiment, we selected for only a single user in a single room in a single location with a single orientation, the details of which are seen in Table 5.1a, with a 70-30 split for

Criteria	Value
User	1
Room	1
Torso location	1
Orientation	1

(a) Single-domain split criteria.

Criteria	Training Split	Testing Split
User	3, 5, 10, 11, 12	1
Room	1	1
Torso location	1, 2, 3, 4, 5	1, 2, 3, 4, 5
Orientation	1	1

(b) Single user leave out criteria.

Table 5.1: Selection criteria for samples in our experiments with the Widar 3.0 dataset.

training-testing. As input, we provide our model with both the BVP as well as the CSI, after being passed through our pipeline discussed in Section 4.3 and the GAF transformation.

This resulted in the performance seen in Table 5.2. Our results here show that DARLInG is in fact capable of performing gesture classification and provides us the confidence to continue with our experiments.

5.1.2 Hyperparameter Optimization

To ensure that our experiments are being run with the optimal hyperparameters, we perform hyperparameter tuning using Weights and Biases and their sweep feature [4]. We setup our sweep to use Bayesian Optimization, the details of which are out of the scope of this thesis but can be found in the author's bachelors thesis [29]. We utilize hyperparameter tuning extensively, running a total of 817 runs with each run running for 50 epochs. The results of the hyperparameter tuning can be seen in Table 5.3 and was used for all further experiments.

Input modality	Accuracy	Precision	F1-Score
BVP	92.22%	90.91%	**0.9655**
CSI	90.23%	90.91%	**0.9120**

Table 5.2: DARLInG's VAE component's performance on gesture prediction on a single-domain split of the Widar 3.0 dataset.

Parameter	Possible values	Value distribution	Optimized value
	RL Embedding Agent		
Start Epoch	$[0, 50]$	Int Uniform	15
	Encoder		
Activation Function	{Leaky ReLU, SeLU, ReLU}	Uniform	Leaky ReLU
Dropout	$[0, 0.9]$	Uniform	0.18
Initial Kernel Size	$[3, 12]$	Int Uniform	7
Latent Variables	$[10, 100]$	Int Uniform	60
Num Conv Blocks	$[1, 10]$	Int Uniform	3
	Multi-task Module		
Decoder Activation Function	{Leaky ReLU, SeLU, ReLU}	Uniform	Leaky ReLU
Decoder Dropout	$[0, 0.9]$	Uniform	0.26
Classifier Activation Function	{Leaky ReLU, SeLU, ReLU}	Uniform	SeLU
Classifier Dropout	$[0, 0.9]$	Uniform	0.28
Classifier num layeres	$[1, 10]$	Uniform	4
	Optimizer		
α	$[1 \times 10^{-10}, 1]$	log uniform	8×10^{-5}
Learning rate	$[1 \times 10^{-6}, 1 \times 10^{-2}]$	uniform	2.2×10^{-4}
Optimizer	{SGD, Adam}	Uniform	Adam

Table 5.3: Hyperparameter tuning results. Each section covers a different grouping of hyperparameters affecting a different component or module of DARLInG.

5.2 Experimental Setup and Results

To answer our research questions, we set up a series of experiments with single-user leave out validation. The criteria used to choose which samples were used as training samples and which for testing are shown in Table 5.1b and resulted in a $\frac{2}{3} : \frac{1}{3}$ split of the data. We additionally held $\frac{1}{10}$ of the training data as a hold-out split to test in- vs. out-of-domain performance of our model. We chose these specific users due to them all being males of similar BMI, as seen in Figure 12 of [46]. Using this data, we set up experiments to answer our research questions from Section 1.3. These experiments ran every combination of RL agent (PPO, DDPG, and known domain factors as a baseline), signal-to-image transform (GAF, MTF, and RP), and encoding method (probability-measure and one-hot) that we have previously chosen.

The results of our experimental runs can be seen in Table 5.4 and are visualized in Figure 5.1. This table, and every figure with experimental results, shows the final model performance on the test set. In this table, we show the performance of DARLInG with two RL algorithms: PPO and DDPG, as well as when the embed head is provided the known domain factors. For each embedding type, we encode the output as either a probability measure (PM) or as a one-hot encoded vector. Finally, for each algorithm and encoding, we provide an input image of the CSI as transformed by GAF, MTF, and RP, details of which can be found in 3.5.

Another way to analyze our runs is to investigate the difference in performance of our model on in-domain vs. out-of-domain samples. To do so, we ran our trained models on the held-out split of the training set. The results can be seen in Figure 5.2. These results indicate that in all cases, our model performs better on in-domain samples, as is expected. The performance on out-of-domain samples are much worse on the known encoding, indicating that the known encoding of domain factors, while usable, is not great for out-of-domain samples. Our results also support that DDPG performs best for out-of-domain samples.

The performance of the embed heads compared to their respective null heads can be seen in Figure 5.3, in which greater positive values indicate that the embed head is performing better than the null head. We can see that with MTF and RP, DDPG out-performs PPO while PPO slightly outperforms DDPG with GAF. Using known domain factors improves performance, but only with GAF and RP transforms while it has the decreases performance with MTF. The largest increase in performance can be seen with MTF transforms and the DDPG RL algorithm with one-hot encoding. Interestingly, PPO with RP seems to provide no actual benefit.

To answer our second research question, we compare the performance of a domain em-

			Accuracy		Precision		F1-Score	
Transform	RL Agent	Encoding	Null head	Embed head	Null head	Embed head	Null head	Embed head
GAF	Known	-	41.0%	42.7%	44.6%	44.6%	0.402	**0.439**
GAF	PPO	PM	33.9%	35.7%	33.9%	35.7%	0.339	**0.357**
GAF	PPO	One-hot	36.6%	38.4%	36.6%	38.4%	0.339	**0.354**
GAF	DDPG	PM	37.1%	36.7%	38.4%	38.4%	0.364	**0.378**
GAF	DDPG	One-hot	42.9%	43.8%	42.0%	40.2%	0.429	**0.438**
MTF	Known	-	37.4%	36.5%	30.4%	29.5%	0.304	**0.295**
MTF	PPO	PM	28.6%	32.1%	28.6%	32.1%	0.369	**0.384**
MTF	PPO	One-hot	46.4%	44.6%	46.4%	44.6%	0.384	**0.393**
MTF	DDPG	PM	37.8%	39.1%	40.2%	42.0%	0.371	**0.395**
MTF	DDPG	One-hot	42.0%	49.1%	42.0%	49.1%	0.376	**0.438**
RP	Known	-	37.5%	38.4%	37.5%	38.4%	0.375	**0.384**
RP	PPO	PM	32.5%	33.1%	33.9%	34.8%	0.328	**0.326**
RP	PPO	One-hot	33.0%	33.2%	30.4%	34.8%	0.331	**0.332**
RP	DDPG	PM	41.1%	42.9%	41.1%	42.9%	0.379	**0.419**
RP	DDPG	One-hot	39.2%	39.7%	37.5%	38.4%	0.389	**0.404**

Table 5.4: Final experimental results of the performance of DARLInG by type of RL agent, domain embedding encoding, and signal-to-image transformation. Each pair of columns represents the results of the null head and the results of the embed head in each of the metrics accuracy, precision, and F1-score. The final F1-Score of the evaluation is emphasized in bold type.

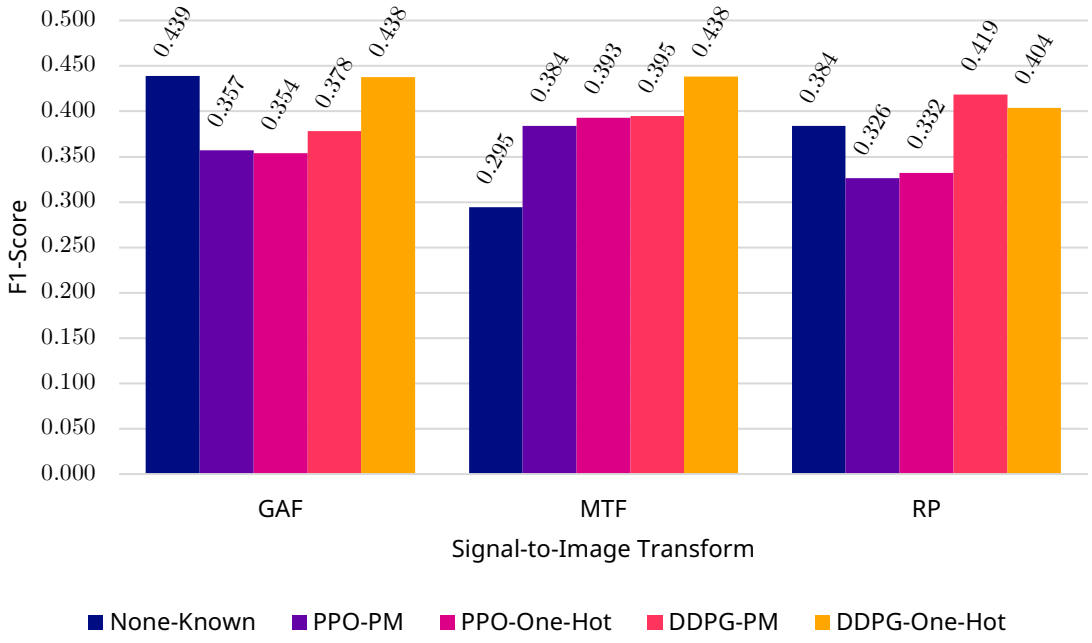


Figure 5.1: The summarized results of the experiments, showing F1-score grouped by the signal-to-image transform used. For each signal-to-image transform, the colors represent the RL algorithm and the encoding used.

bedding encoded as a probability-measure and one using one-hot encoding, the results of which are seen in Figure 5.4. The results show that generally, the encoding method does not make a significant difference for PPO and only makes a difference for DDPG with MTF and RP transforms. With the MTF transform, one-hot is the preferred encoding while in RP, the probability-measure is the prefered encoding.

Comparing all these results with the baseline of a known domain encoding, we see that although the known-domain encoding performs well with GAF and RP, and poorly on MTF, the known-domain encoding fails to provide an adequate representation of out-of-domain samples, as seen in Figure 5.2. This may simply be to the discrete (i.e., categorical) encoding of the domain factors provided by the known domain encoding, instead of the more complex domain representations provided by the RL agents. For example, although subject height is a continuous variable which can be properly encoded by a multi-dimensional one-hot encoding or a probability measure, a one-hot user encoding provides no further information about how height varies between two users. This suggests that the RL agents provide a better encoding across domains, but fall short of hand-crafted domain labels with ground-truth information and no out-of-domain samples.

The results are generally unimpressive, though, with F1-scores significantly below both

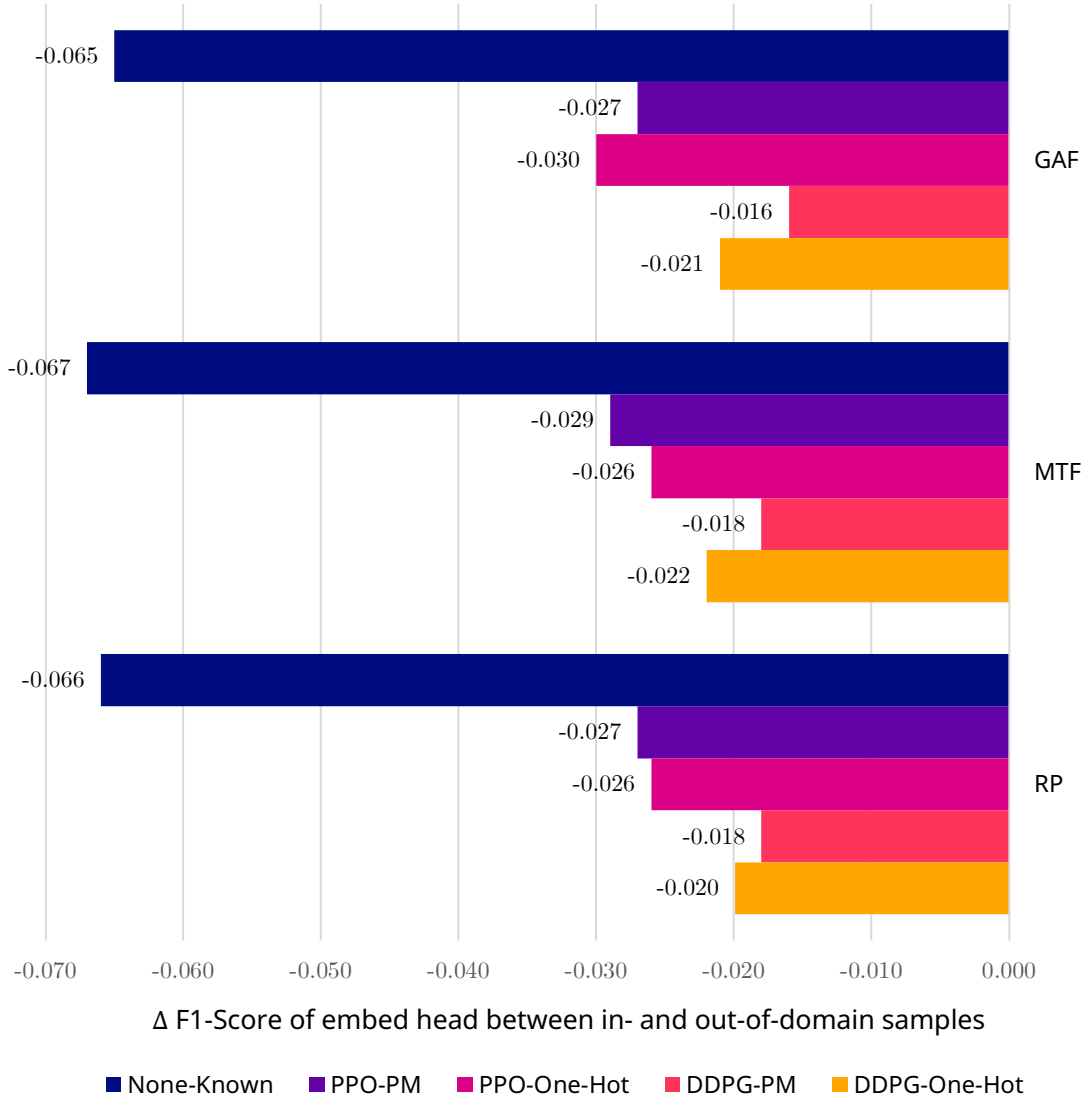


Figure 5.2: The difference in performance of our models on in-domain vs. out-of-domain samples. The in-domain samples were taken from a held-out set of the training samples. The values are grouped by signal-to-image transform and the colors represent the RL algorithm and the encoding used. The shown values indicate the reduction in F1-score of our model on out-of-domain samples w.r.t. the in-domain samples with lower values indicating better performance on the in-domain samples.

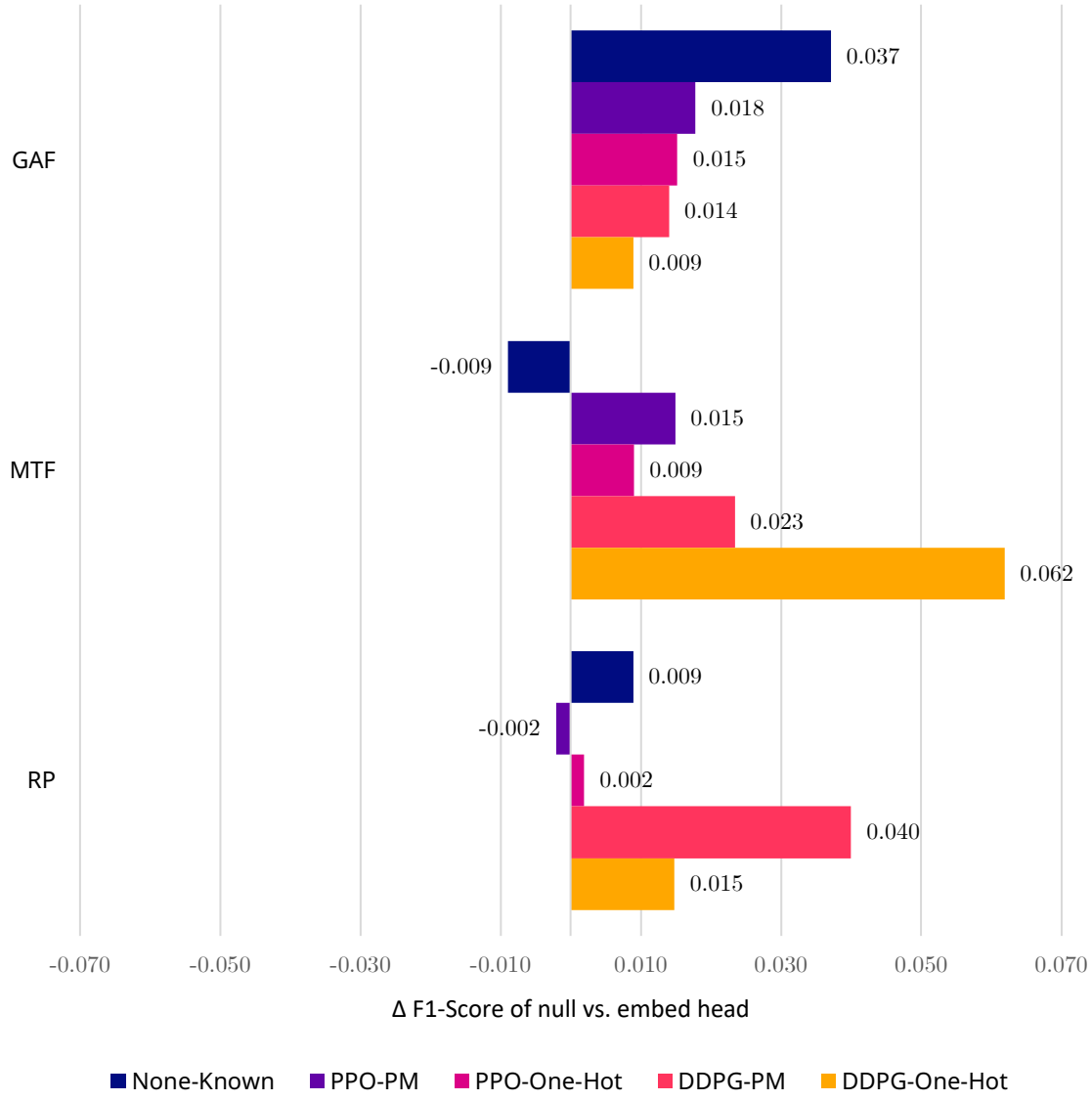


Figure 5.3: This chart shows the performance delta between the null head and the embed head for each experiment. Higher positive values indicate that the embed head achieved a higher F1-score.

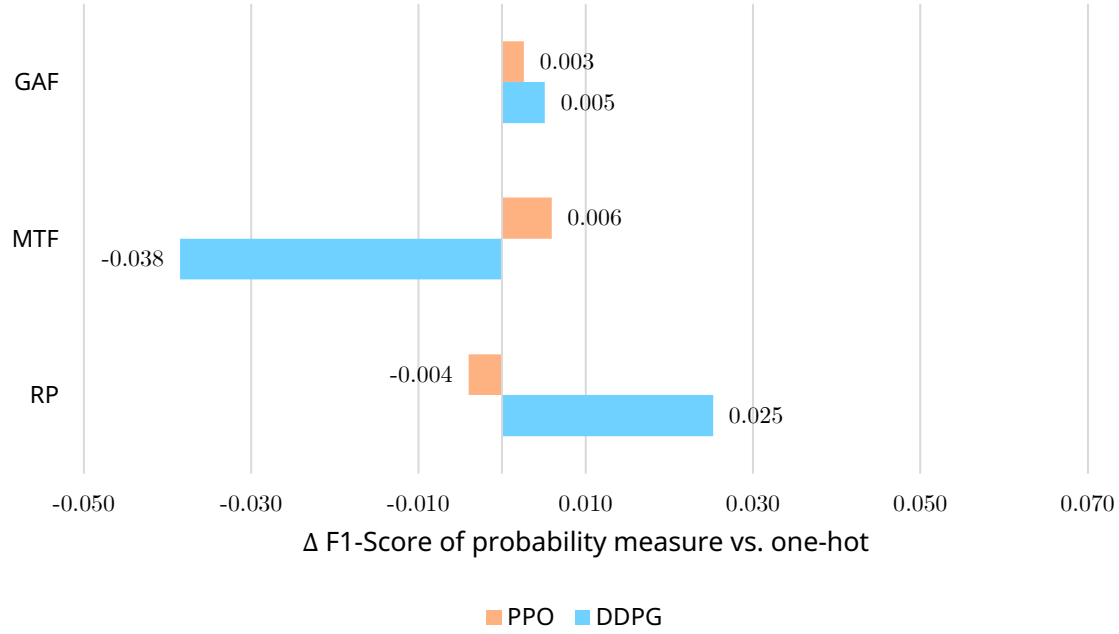


Figure 5.4: This chart shows the overall performance delta between using a probability measure and a one-hot encoding for the domain embedding. The values are grouped by signal-to-image transform and the colors represent the RL algorithm used. The scores used here have also been corrected for their respective algorithm's null head performance, i.e., uses the scores seen in Figure 5.3 instead of in Table 5.4. Higher positive values indicate the embed head achieved a higher F1-score when using the probability measure encoding as opposed to using a one-hot encoding.

	F1-Score
Standard	0.438
Extended	0.553

Table 5.5: Extended training run results of the DDPG agent with MTF transform and one-hot encoding.

the baseline in [46] and any of the state-of-the-art methods discussed in Chapter 2.

5.2.1 Extended Run

As one final experiment, we ran our best performing configuration, i.e., DDPG with MTF transform and one-hot encoding, on the TU/e High-Performance Computing cluster with 100 model epochs and 50400 agent iterations per epoch. The number of agent iterations were chosen as the closest full dataset iterations to 50000 steps, resulting in a total of 3528000 total agent steps as we set the agent training starting epoch $\zeta = 30$. We double ζ to 30 as we also double the total number of epochs during training. This is twice the training length of the VAE and 100$\times$ the training length for the agent relative to the other training runs.

We observe an increase in performance, with an increase in F1-score of 0.12 and the performance plateauing around epoch 75. Although this is a welcome increase in performance, this is still below the baselines in [46] and any of those discussed in Chapter 2. The performance plateauing after epoch 75 does suggest that increasing training duration any further will most likely not increase model performance.

Chapter 6

Discussion

We have shown that our approach is capable of generally improving F1-scores in Widar, but our overall performance is disappointing, compared to both baselines and state-of-the-art approaches. In this chapter, we discuss the experimental results and what these results mean with respect to our research questions, analyze the latent space and domain embeddings produced by our approach, as well as suggest future research that may be taken.

6.1 Comparison to State-of-the-Art

It is difficult to compare our results to the state-of-the-art (SOTA), as we did not train on the same training splits as those SOTA models. We can, though, make a rough comparison to the CNN baseline in the Widar 3.0 paper [46] and to the WiGAN approach in [16]. Both these papers provide results for single-user leave-out and the performance comparison with our method can be seen in Figure 6.1. In the Widar 3.0 paper, the CNN approach used is able to provide a baseline result of 88% accuracy in single-user leave-out while WiGAN is able to achieve 91% accuracy. These methods both provide significantly better results than DARLInG, which achieves a best performance of 55% accuracy when using DDPG with MTF.

It should be reiterated, though, that these numbers are not directly comparable as they do not use the same dataset splits.

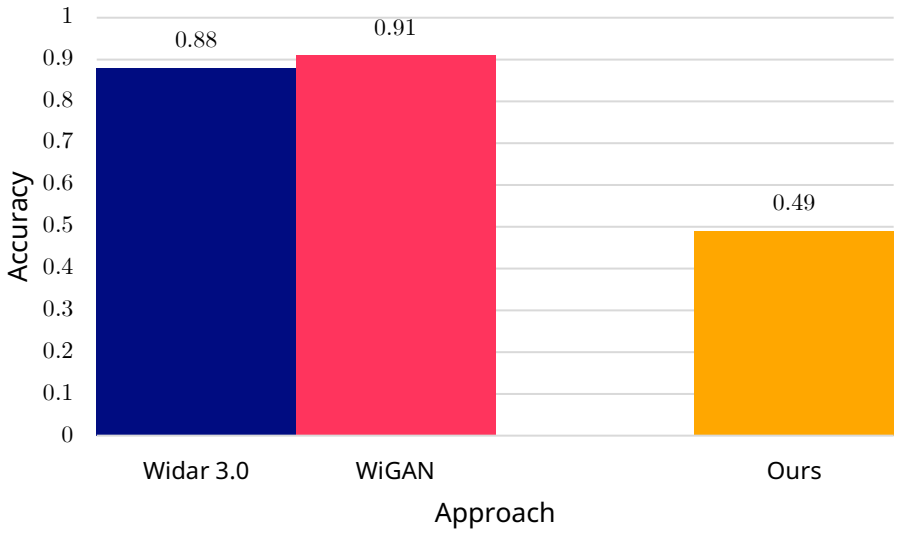


Figure 6.1: Comparison of our method against the Widar 3.0 benchmark and the current SOTA. The space between our work and other works is intentionally placed to reiterate the point that these values are not directly comparable as our data splits, although both representing single-user leave-out, are not comparable.

6.2 Research questions

In this section, we answer each of the research questions we posited in Section 1.3 as well as a short discussion on the strengths and limitations of our approach to answering these questions.

Research Question 1 Our first question asks how well DARLInG's embed heads perform compared to its null heads. The results from Table 5.4 indicate that we were able to achieve up to 7.1 percentage points of improvement on accuracy and an improvement on the F1-score of 0.062 when comparing the null head and the embed head on out-of-domain results. This was found when using MTF as the transform, DDPG as the RL agent, and one-hot encoding as the domain embedding encoding. The results when run with other configurations indicate that the magnitude of the performance increase may not be as large, although it is generally positive.

The only configuration where there was a performance decrease was with PPO using the RP transform. With this configuration, the δ F1-score was -0.002 for the PM encoding and +0.002 for the one-hot encoding. As such, we conclude that this was a negligible difference and should therefore be interpreted as DARLInG not having any effect on model

performance.

We therefore conclude that our results suggest using DARLInG to provide unsupervised domain labels provide similar or better out-of-domain performance on Widar.

Research Question 2 Our second question asks how changing the domain embedding encoding affects performance. Generally, DDPG seems to perform better when using a PM encoding when the transform is RP while it performs better with one-hot encoding when the transform is MTF. For all other combinations, the results in Figure 5.4 show no significance difference in performance between encodings. As a result, we believe that the domain embedding is not significant and that other factors contribute more towards the performance of the model. Further research may be necessary to investigate the optimal encoding of the domain embedding.

Research Question 3 Our last research question asks how changing the signal-to-image transformation affects performance. Our results indicate that generally, GAF performs consistently, but not impressively. MTF seems to perform quite well with DDPG and similarly to GAF with PPO. RP seems to perform poorly with PPO, not improving over the null head, while performing similarly to MTF with DDPG. As a result, we can confidently conclude that different signal-to-image transformations affect model performance, but the results are inconclusive as to which transformation works best. The results do suggest, though, that MTF may be the best performer in our experiments of limited sample size.

Strengths and Limitations We believe that our experiments are quite thorough with significant resources put into ensuring that the model itself is working appropriately, as well as that the code is free of any significant bugs which could render the results invalid. Significant effort was put into ensuring that every step of the CSI processing and transformation pipeline produced the correct results during the course of writing the thesis. The VAE was also checked by running the VAE with the Fashion-MNIST dataset [42]. This produced results as expected with the VAE being able to adequately reconstruct the input images as well as classify the images appropriately with > 0.85 accuracy, matching the benchmarks in [42]. Finally, the RL algorithms used are off-the-shelf implementations from Stable Baselines 3 [27] implemented according to the sample code provided, reducing the likelihood of issues in the code with the RL algorithms. We also approached a PhD candidate at the TU/e specializing in RL who concurred with our ideas on the alternating training style, where we iterate between training the RL agent and the VAE [13].

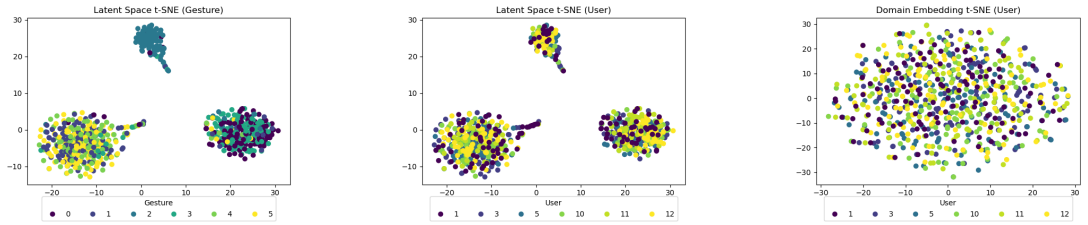
Limitations include reduced hyperparameter tuning of the RL agents. RL agents are notoriously time-consuming to train [30, 31, 20]. With our limited compute resources, we were unable to conduct a full hyperparameter optimization sweep of the RL agents. We used known good hyperparameters provided in the documentation of [27] for dealing with a continuous action space for our agents.

6.3 Latent Space and Domain Embedding Analysis

To properly analyze the low performance of our model, we would like to analyze both the latent space $\boldsymbol{z}$ produced by our encoder as well as the domain embeddings $\boldsymbol{d}_r$ produced by our RL agent. We do this by running the entire dataset through both the encoder and RL agent and produce t-SNE embedding plots with their outputs. By doing so, we can see if there is any structure to the embeddings. We specifically analyze the case of using MTF as the signal-to-image transformation, PPO as the agent, and both the one-hot and probability measure domain embedding encodings, the results of which can be seen in Figures 6.2 and 6.3.

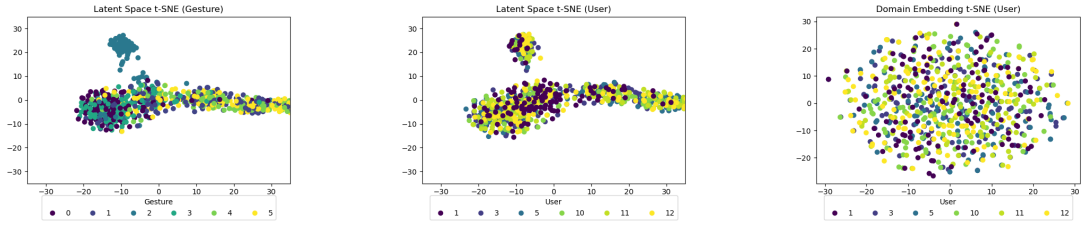
We can see that for the latent space, there is a much clearer structure in both configurations, especially with gesture 2 being separated out quite distinctly in both Figures 6.2a and 6.3a. Interestingly, both Figures 6.2c and 6.3c show quite an even distribution of users throughout each cluster in the t-SNE space.

For the domain embeddings, we see that there is *some* structure in Figures 6.2c and 6.3c, as it is not only a single normally-distributed disk, but there is not much of it. We also see that the users, including the single-leave out user (User 1) is distributed quite evenly throughout the space in both figures. This suggests that the RL agent is not able to accurately provide significantly different domain embeddings between the users. We believe



(a) t-SNE by gesture (b) t-SNE by user (c) t-SNE by User

Figure 6.2: t-SNEs of the latent space and domain embeddings produced by PPO with one-hot encoding and MTF transformation.



(a) t-SNE by gesture (b) t-SNE by user (c) t-SNE by user

Figure 6.3: t-SNEs of the latent space and domain embeddings produced by PPO with PM encoding and MTF transformation.

this may be simply due to training time constraints.

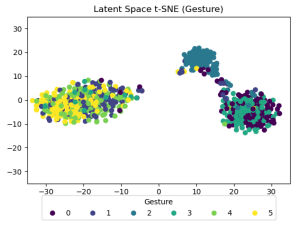
To test this hypothesis, we trained our agent for an extended period of time, as was described in Subsection 5.2.1. The t-SNEs of this run can be seen in Figure 6.4.

We notice some clearer separation by gesture in Figure 6.4a, with gestures 2 and 0 being more clearly separated on the right side clusters. Additionally, Figure 6.4b shows some more clustering within clusters for the users. For example, user 1 is primarily clustered towards to the top of the left-side cluster, for example. Finally, we finally see some structure brought to the domain embeddings, with 3 clearly visible clusters in Figure 6.4c, although they do not seem to correspond to the users themselves.

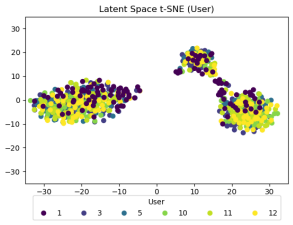
We additionally plotted the domain embedding t-SNE colored by gesture in Figure 6.4d, and we notice that the domain embeddings are instead clustered by gesture. This is interesting, as it means that our domain embeddings are in fact directly providing data on the gesture for the embed head, instead of simply providing data about the domain. This would imply that we have some sort of meta-model made up of the embed head and the RL agent which together attempt to provide better gesture classifications.

We also plotted the domain-embedding t-SNE colored by gesture for both short run configurations, but we see no such clustering in those and as such, for brevity, will not include the plots.

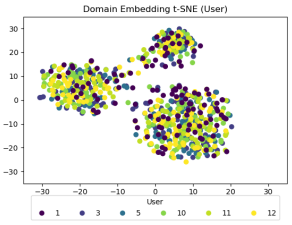
We also plotted the latent-space and domain embedding t-SNEs, filtered by user and colored by gesture to analyze whether this provides any additional insight. After analyzing these plots, we do not see any additional insight that can be taken from them, but nonetheless they are available in Appendix B.



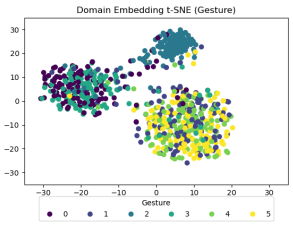
(a) t-SNE by gesture



(b) t-SNE by user



(c) t-SNE by user



(d) t-SNE by gesture

Figure 6.4: t-SNEs of the latent space and domain embeddings produced by PPO with one-hot encoding and MTF transformation with extended training time.

6.4 Future Research

We believe that there may still be potential in our approach. Further research could be aimed at a different neural-network architecture, tuning of the RL agents and some changes to the training regime.

6.4.1 A Different Architecture

Although our preliminary experiments in Section 5.1 suggested that our VAE model could perform adequately for the task of gesture classification with Widar, it was evidently not up to the task of gesture classification where the input is domain shifted. One may, instead of using a VAE, use one of the known performant models from Section 2.2. This may result in better cross-domain performance to begin with with our RL approach adding the few percentage-points on top to make said models competitive with single-domain models.

6.4.2 Tuning of the RL Agent

Due to time constraints, we conducted little hyperparameter tuning of the RL agent due to time and compute constraints. Future research may simply repeat our experiments but with increased time and compute to see if that could help at all.

We might also develop a different reward function which forces the agent to provide a domain embedding which more clearly provides additional information about the domain itself, instead of additional information about the gesture as our agent seems to have done as seen in the analysis in Section 6.3.

6.4.3 A Different Training Regime

Finally, we see in the loss curves of our experiments that there is a always a sudden increase in the loss function of the model, with a corresponding drop of around 0.05—0.10 in validation accuracy of the null head, at the epoch in which we introduce the embed head. This decrease in validation accuracy of the null head may indicate that the changes our model makes to the latent space while trying to optimize for the embed head and RL agent is adversely affecting the performance of the null head. Put another way, the null head is not capable of adapting to the changes in the latent space which the model makes while trying to optimize for the embed head. We suggest that future work could look into two different approaches to this problem: By training the embed head as a completely separate model on a frozen encoder, or by changing the way we train the embed head using a process we propose called Simulated Aging and Hardening.

It might be possible to reduce the decrease in performance of the null head, and potentially increase overall performance, by training the embed head as an entirely separate network to the encoder and null heads. In this approach, during the VAE training portion of every epoch, the encoder and null head would first be trained over the entire training set. After every batch has been processed and the optimizers run for the encoder and null head, the parameters of said modules would be frozen. At this point, we will start optimization on the embed head. The same training set will be run through the encoder again and the produced outputs $\boldsymbol{z}$ along with the produced domain embeddings $\boldsymbol{d}_r$ from the RL agent (also frozen) would be run through the embed head to produce its gesture class predictions. This would ensure that the embed head is being trained to predict only based on the $\boldsymbol{z}$ and $\boldsymbol{d}_r$ without affecting the encoder or the null head. Then finally, with everything frozen, we train only the RL agent. By doing so, the RL agent is forced to learn from an environment which is not itself adapting to the actions of the agent, potentially simplifying the environment evolution for the agent.

We might also attempt to instead slowly increase the rate of optimization of the embed head in a process we propose to be called *Simulated Aging and Hardening* (SAH), directly inspired by the analogy used by simulated annealing (SA) to metallurgy. In SA, we use an initial solution and probabilistically explore neighboring solutions with exploratory probability.

Over time, we slowly "cool down" the probability of accepting worse solutions until we arrive at a near-optimal solution. In this way, SA can be said to start of very exploratory and slowly reduce its exploration and increase its exploitation of known good solutions. Conversely, our proposed SAH follows the process of aging and hardening, a process in metallurgy with similar results to annealing but with opposite processes. The steps to hardening in metallurgy is typically as follows: Start with a metal that is at a high temperature, quench it to rapidly cool it down "freezing" the alloying elements in the metal, and finally slowly heat up the metal in a process called aging and hardening.

In SAH, applied directly to our use case, we would train the encoder and null heads in the same way we do now until the epoch where the embed head and RL agent starts training. At this point, we would freeze the training of the encoder and null heads. The embed head is then initialized with a copy of the parameters of the null head. As the parameters are equivalent, it is unlikely to be able to process input from the RL agent properly. We start off by using a low learning rate on the embed head to avoid sudden large changes to its parameters and avoiding instability while looking for the new minima. After a certain number of epochs, or some heuristic is reached, we then continue training the model as a whole with the current training method described in this thesis.

Chapter 7

Conclusions

In this thesis, we have presented DARLInG (Domain Auto-labeling through Reinforcement Learning for the Identification of Gestures), our approach to domain-shift mitigation using unsupervised domain label generation through reinforcement learning. Our model incorporates a Variational Autoencoder as the main encoder-decoder-classifier model while testing both Deep Deterministic Policy Gradient and Proximal Policy Optimization as the Reinforcement Learning agent for domain label generation. As our signal-to-image transformation, we have investigated the use of Gramian Angular Fields, Markov Transition Fields, and Recurrent Plots. We have tested this approach on a single-user leave-out split of the Widar 3.0 [46] dataset.

We have ran 15 different configurations of our approach, with 12 of these utilizing an RL agent as a means for domain label generation. In these configurations, the best result was using DDPG as the agent, MTF as the signal-to-image transformation, and one-hot as the domain embedding encoding. Our results are inconclusive, but indicate that there *may* be some promise to our approach. What results we do have, though, show that DARLInG does generally increase performance, or at least does not negatively impact performance, over a standard VAE's performance at cross-domain gesture recognition. We then analyzed why our approach may have not worked and suggest directions for future research which may be able to turn DARLInG into a viable approach for domain shift migitagion through unsupervised domain label generation. As such, we conclude that, DARLInG as an approach to domain-shift mitigation, may have potential, although future research will be necessary to truly investigate whether DARLInG can truly be the darling approach its authors had originally hoped for.

Bibliography

[1] Fadel Adib and Dina Katabi. “See through walls with WiFI’. In: *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*. 2013, pp. 75–86.

[2] Fadel Adib et al. “3D tracking via body radio reflections”. In: *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*. 2014, pp. 317–329.

[3] Bram van Berlo, Tanir Ozcelebi, and Nirvana Meratnia. “Insights on mini-batch alignment for wifi-csi data domain factor independent feature extraction”. In: *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. IEEE. 2022, pp. 527–532.

[4] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.

[5] Lieke A.J. van den Biggelaar. “Gesture recognition using Deep Q-Neworks on WiFi-CSI data”. MA thesis. Eindhoven University of Technology, 2022.

[6] Tom B. Brown et al. “Language Models are Few-Shot Learners”. In: *CoRR* abs/2005.14165 (2020). arXiv: 2005.14165. URL: https://arxiv.org/abs/2005.14165.

[7] Kevin Chetty, Graeme E Smith, and Karl Woodbridge. “Through-the-wall sensing of personnel using passive bistatic wifi radar at standoff distances”. In: *IEEE Transactions on Geoscience and Remote Sensing* 50.4 (2011), pp. 1218–1226.

[8] Rui Du et al. “An overview on IEEE 802.11 bf: WLAN sensing”. In: *arXiv preprint arXiv: 2207.04859* (2022).

[9] Jean-Pierre Eckmann, S Oliffson Kamphorst, David Ruelle, et al. “Recurrence plots of dynamical systems”. In: *World Scientific Series on Nonlinear Science Series A* 16 (1995), pp. 441–446.

[10] Technical University of Eindhoven. *Atlas most sustainable education building in the world Atlas most sustainable education building in the world Atlas most sustainable education building in the world*. Apr. 2019. URL: https://www.tue.nl/en/our-university/

tue-campus/buildings/atlas/atlas-most-sustainable-education-building-in-the-world/.

[11] Johann Faouzi and Hicham Janati. "pyts: A Python Package for Time Series Classification". In: *Journal of Machine Learning Research* 21.46 (2020), pp. 1–6. URL: http://jmlr.org/papers/v21/19-763.html.

[12] Varun Godbole et al. *Deep Learning Tuning Playbook*. Version 1.0. 2023. URL: http://github.com/google/tuning_playbook.

[13] Bram Grooten. *Personal Interview*. Personal Interview. Mar. 2023.

[14] Mohammad Hasan. *State of IoT 2022 Number of connected IoT devices growing 18% to 14.4 billion globally*. URL: https://iot-analytics.com/number-connected-iot-devices/ (visited on 05/18/2022).

[15] Wenfeng He et al. "WiG: WiFi-based gesture recognition system". In: *2015 24th International Conference on Computer Communication and Networks (ICCCN)*. IEEE. 2015, pp. 1–7.

[16] Dehao Jiang, Mingqi Li, and Chunling Xu. "Wigan: A wifi based gesture recognition system with gans". In: *Sensors* 20.17 (2020), p. 4757.

[17] Wenjun Jiang et al. "Towards environment independent device free human activity recognition". In: *Proceedings of the 24th annual international conference on mobile computing and networking*. 2018, pp. 289–304.

[18] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[19] Diederik P. Kingma and Max Welling. "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392. DOI: 10.1561/2200000056. URL: https://doi.org/10.1561%2F2200000056.

[20] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[21] Yongsen Ma et al. "Location-and person-independent activity recognition with WiFi, deep neural networks, and reinforcement learning". In: *ACM Transactions on Internet of Things* 2.1 (2021), pp. 1–25.

[22] Yongsen Ma et al. "Signfi: Sign language recognition using wifi". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2.1 (2018), pp. 1–21.

[23] Mauro Martini et al. “Domain-Adversarial Training of Self-Attention-Based Networks for Land Cover Classification Using Multi-Temporal Sentinel-2 Satellite Imagery”. In: *Remote Sensing* 13.13 (2021). ISSN: 2072-4292. DOI: 10.3390/rs13132564. URL: https://www.mdpi.com/2072-4292/13/13/2564.

[24] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *nature* 518.7540 (2015), pp. 529–533.

[25] C.H.J.M Oerlemans. “The Effect of Data Preprocessing On the Performance of Few-shot Learning for Wi-Fi CSI- based Gesture Recognition The Effect of Data Preprocessing on the Performance of Few-shot Learning for Wi-Fi CSI-based Gesture Recognition”. MA thesis. Tilburg University, 2022.

[26] Qifan Pu et al. “Whole-home gesture recognition using wireless signals”. In: *Proceedings of the 19th annual international conference on Mobile computing & networking*. 2013, pp. 27–38.

[27] Antonin Raffin et al. “Stable-Baselines3: Reliable Reinforcement Learning Implementations”. In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html.

[28] Yvan Putra Satyawan. “CNNs and Preprocessing: The Dynamic Duo of Wi-Fi Localization”. Jan. 2023.

[29] Yvan Putra Satyawan. *Semantic Segmentation of Curbs and Curb Cuts from Street Imagery*. Bachelor’s Thesis. Sept. 2019.

[30] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv: 1707.06347* (2017).

[31] John Schulman et al. *Trust Region Policy Optimization*. 2017. arXiv: 1502.05477 [cs.LG].

[32] Signify B.V. *Groundbreaking motion detection turns your lights on and off without the need for sensors*. Press Release. Sept. 2022. URL: https://www.signify.com/global/our-company/news/press-releases/2022/20220916-signify-delivers-new-way-to-automate-your-lights-using-wi-fi-sensing-technology (visited on 08/31/2023).

[33] Signify B.V. *Smart lighting for your daily living: WiZ launches a brand new generation of products in Europe*. Aug. 2020. (Visited on 08/31/2023).

[34] Suze E. Sips. “Impact analysis of network pruning and quantization on the domain shift problem in a recognition context, using WiFi CSI data”. MA thesis. Eindhoven University of Technology, 2022.

[35] Maryam Tavakol. *Data Intelligence Challenge*. Lecture. Apr. 2022.

[36] Tesla Inc. *Data Sharing End User License Agreement*. EULA. May 2017.

[37] Mark Towers et al. *Gymnasium*. Mar. 2023. DOI: 10.5281/zenodo.8127026. URL: https://zenodo.org/record/8127025 (visited on 07/08/2023).

[38] Lukas Tuggener et al. "The DeepScoresV2 dataset and benchmark for music object detection". In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 9188–9195.

[39] Fei Wang et al. "Person-in-WiFi: Fine-grained person perception using WiFi". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 5452–5461.

[40] Zhiguang Wang and Tim Oates. "Imaging time-series to improve classification and imputation". In: *24th International Joint Conference on Artificial Intelligence*. 2015.

[41] Christopher John Cornish Hellaby Watkins. "Learning from delayed rewards". In: (1989).

[42] Han Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms". In: *arXiv preprint arXiv:1708.07747* (2017).

[43] Hongfei Xue et al. "DeepMV: Multi-view deep learning for device-free human activity recognition". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4.1 (2020), pp. 1–26.

[44] Weidong Zhang, Zexing Wang, and Xuangou Wu. "WiFi signal-based gesture recognition using federated parameter-matched aggregation". In: *Sensors* 22.6 (2022), p. 2349.

[45] Youshan Zhang, Hui Ye, and Brian D Davison. "Adversarial reinforcement learning for unsupervised domain adaptation". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 635–644.

[46] Yue Zheng et al. "Zero-effort cross-domain gesture recognition with Wi-Fi". In: *Proceedings of the 17th annual international conference on mobile systems, applications, and services*. 2019, pp. 313–325.

[47] Andrii Zhuravchak, Oleg Kapshii, and Evangelos Pournaras. "Human Activity Recognition based on Wi-Fi CSI Data-A Deep Neural Network Approach". In: *Procedia Computer Science* 198 (2022), pp. 59–66.

[48] Augustinas Zinys, Bram van Berlo, and Nirvana Meratnia. "A domain-independent generative adversarial network for activity recognition using wifi csi data". In: *Sensors* 21.23 (2021), p. 7852.

Appendix A

Dataset Analysis

We explored the distribution of domain factors in the dataset. The results of said exploration can be found in this chapter of the appendix.

	Room Number			
User	1	2	3	Total
1	790	225	0	1015
2	550	550	0	1100
3	300	375	150	825
4	0	150	0	150
5	225	150	0	375
6	0	300	0	300
7	0	0	150	150
8	0	0	150	150
9	0	0	150	150
10	225	0	0	225
11	225	0	0	225
12	225	0	0	225
13	225	0	0	225
14	225	0	0	225
15	225	0	0	225
16	225	0	0	225
17	225	0	0	225
Total	3665	1750	600	6015

Table A.1: Distribution of users vs. room number in the dataset. Note that not all users are recorded in all rooms and that the room distribution is usually unbalanced. Room 3 also has the fewest number of samples by an order of magnitude.

	Gestures										
User	1	2	3	4	5	6	7	8	9	10	Total
1	130	130	130	130	130	130	65	65	65	40	1015
2	200	175	175	175	150	125	25	25	25	25	1100
3	150	150	150	125	125	125	0	0	0	0	825
4	25	25	25	25	25	25	0	0	0	0	150
5	50	50	50	50	50	50	25	25	25	0	375
6	50	50	50	50	50	50	0	0	0	0	300
7	25	25	25	25	25	25	0	0	0	0	150
8	25	25	25	25	25	25	0	0	0	0	150
9	25	25	25	25	25	25	0	0	0	0	150
10	25	25	25	25	25	25	25	25	25	0	225
11	25	25	25	25	25	25	25	25	25	0	225
12	25	25	25	25	25	25	25	25	25	0	225
13	25	25	25	25	25	25	25	25	25	0	225
14	25	25	25	25	25	25	25	25	25	0	225
15	25	25	25	25	25	25	25	25	25	0	225
16	25	25	25	25	25	25	25	25	25	0	225
17	25	25	25	25	25	25	25	25	25	0	225
Total	880	855	855	830	805	780	315	315	315	65	6015

Table A.2: Distribution of users vs. gestures performed. Note that gestures 1-6 are always balanced by user, but some users performed more than others. Gestures 7-9 are performed by only some users and gesture 10 is performed by only users 1 and 2.

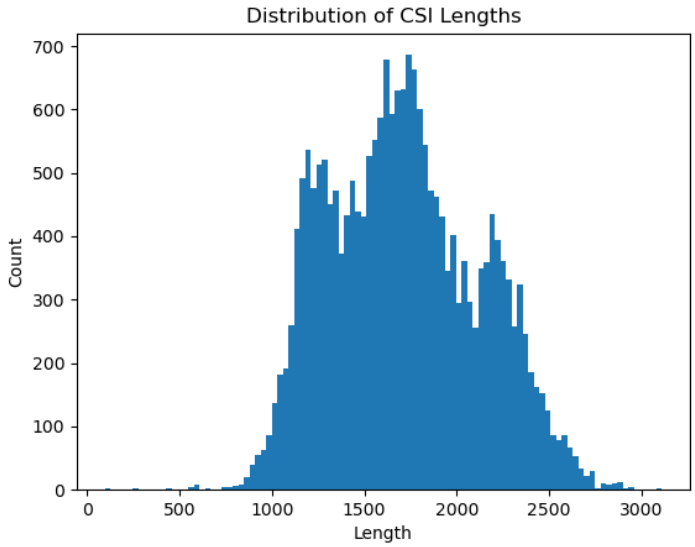


Figure A.1: Histogram of the length of CSI datapoints. CSI datapoints collected were of different lengths with the majority being less than 2000 samples long, representing samples of around 2 seconds each.

	Repetitions			
Room	5	10	20	Total
1	2025	950	690	3665
2	1750	0	0	1750
3	600	0	0	600
Total	4375	950	690	6015

Table A.3: Distribution of rooms vs repetitions. All samples were repeated at least 5 times. Specifically, in room 1, some samples were repeated 10 and 20 times.

	Torso Locations								
User	1	2	3	4	5	6	7	8	Total
1	155	155	155	155	155	80	80	80	1015
2	220	220	220	220	220	0	0	0	1100
3	165	165	165	165	165	0	0	0	825
4	30	30	30	30	30	0	0	0	150
5	75	75	75	75	75	0	0	0	375
6	60	60	60	60	60	0	0	0	300
7	30	30	30	30	30	0	0	0	150
8	30	30	30	30	30	0	0	0	150
9	30	30	30	30	30	0	0	0	150
10	45	45	45	45	45	0	0	0	225
11	45	45	45	45	45	0	0	0	225
12	45	45	45	45	45	0	0	0	225
13	45	45	45	45	45	0	0	0	225
14	45	45	45	45	45	0	0	0	225
15	45	45	45	45	45	0	0	0	225
16	45	45	45	45	45	0	0	0	225
17	45	45	45	45	45	0	0	0	225
Total	1155	1155	1155	1155	1155	80	80	80	6015

Table A.4: Distribution of users vs. torso locations. Torso locations are always balanced, by only user 1 performs in torso location 6-8.

	Room			
Torso Location	1	2	3	Total
1	685	350	120	1155
2	685	350	120	1155
3	685	350	120	1155
4	685	350	120	1155
5	685	350	120	1155
6	80	0	0	80
7	80	0	0	80
8	80	0	0	80
Total	3665	1750	600	6015

Table A.5: Distribution of rooms vs. torso locations. Torso locations are always balanced, but torso locations 6-8 are only performed in room 1. Given that only user 1 performs these locations and the number of samples recorded by user one in Table A.4 is the same as in this table, we can assume only user 1 performed in these locations.

	Face Orientation					
Room, User	1	2	3	4	5	Total
Room 1	**733**	**733**	**733**	**733**	**733**	**3665**
1	158	158	158	158	158	790
2	110	110	110	110	110	550
3	60	60	60	60	60	300
5	45	45	45	45	45	225
10	45	45	45	45	45	225
11	45	45	45	45	45	225
12	45	45	45	45	45	225
13	45	45	45	45	45	225
14	45	45	45	45	45	225
15	45	45	45	45	45	225
16	45	45	45	45	45	225
17	45	45	45	45	45	225
Room 2	**350**	**350**	**350**	**350**	**350**	**1750**
1	45	45	45	45	45	225
2	110	110	110	110	110	550
3	75	75	75	75	75	375
4	30	30	30	30	30	150
5	30	30	30	30	30	150
6	60	60	60	60	60	300
Room 3	**120**	**120**	**120**	**120**	**120**	**600**
3	30	30	30	30	30	150
7	30	30	30	30	30	150
8	30	30	30	30	30	150
9	30	30	30	30	30	150
Total	1203	1203	1203	1203	1203	6015

Table A.6: Distribution of Face orientation by room and user. Face orientations are always balanced by both user and room and all face orientations are always performed, although the total number of performances by user may be different.

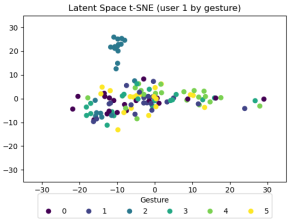
	Repetitions			
Gesture	5	10	20	Total
1	650	115	115	880
2	625	115	115	855
3	625	115	115	855
4	600	115	115	830
5	575	115	115	805
6	550	115	115	780
7	250	65		315
8	250	65		315
9	250	65		315
10		65		65
Total	4375	950	690	6015

Table A.7: Distribution of repetitions vs. gestures. No specific gesture was repeated more often, but only gestures 1-6 were repeated 20 times. Gesture 10 specifically also only had 10 repetitions.

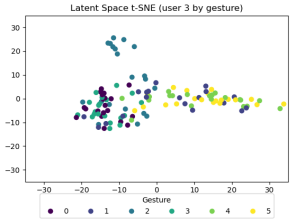
Appendix B

Latent Space and Domain Embedding t-SNE Filtered by User

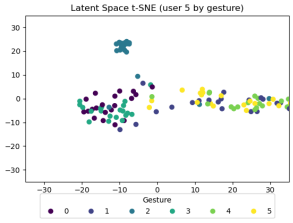
We plotted the latent space and domain embedding t-SNEs with the DDPG agent trained on the MTF transform and one-hot encoding for both the model initially trained and the model that was trained with extended epochs and agent steps. We felt that no additional conclusions could be drawn from these plots and as such did not include them in our main discussion section but include them here for posterity.



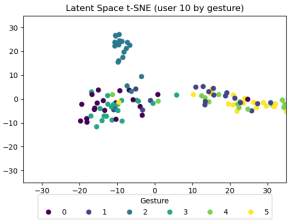
(a) t-SNE by gesture for User 1



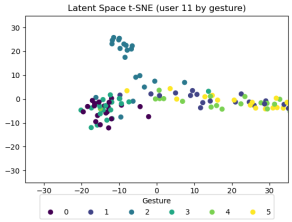
(b) t-SNE by gesture for User 3



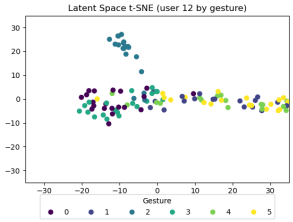
(c) t-SNE by gesture for User 5



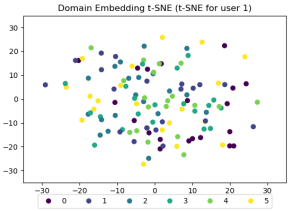
(d) t-SNE by gesture for User 10



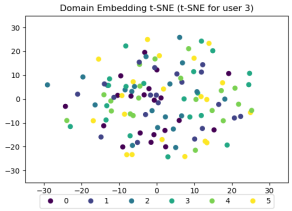
(e) t-SNE by gesture for User 11



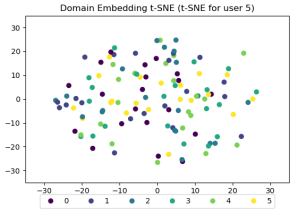
(f) t-SNE by gesture for User 12



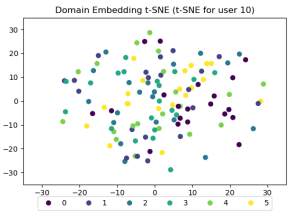
(g) t-SNE by gesture for User 1



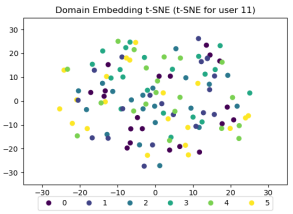
(h) t-SNE by gesture for User 3



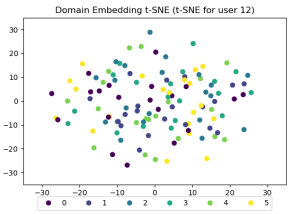
(i) t-SNE by gesture for User 5



(j) t-SNE by gesture for User 10

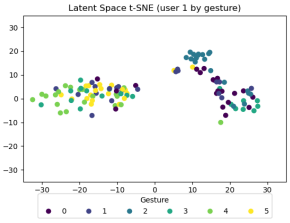


(k) t-SNE by gesture for User 11

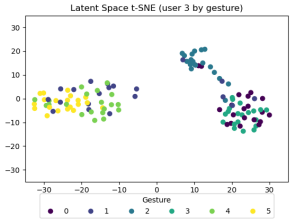


(l) t-SNE by gesture for User 12

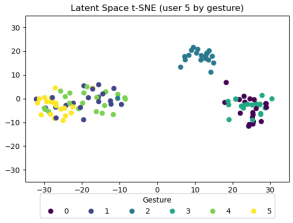
Figure B.1: t-SNEs of the latent space and domain embeddings produced by PPO with one-hot encoding and MTF transformation with the initial training time, filtered by User.



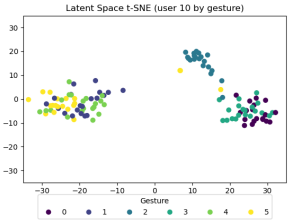
(a) t-SNE by gesture for User 1



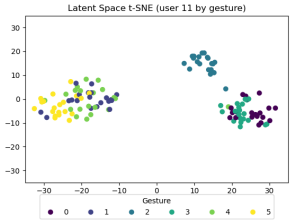
(b) t-SNE by gesture for User 3



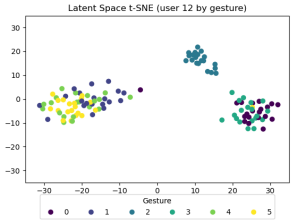
(c) t-SNE by gesture for User 5



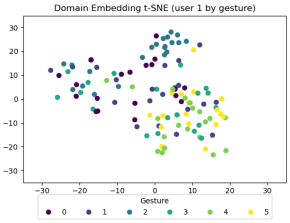
(d) t-SNE by gesture for User 10



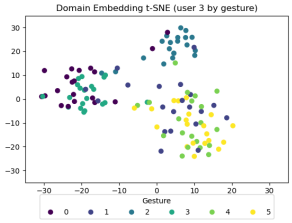
(e) t-SNE by gesture for User 11



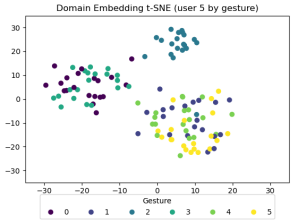
(f) t-SNE by gesture for User 12



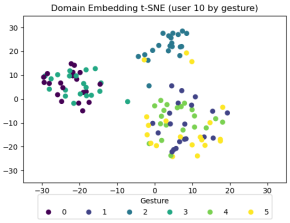
(g) t-SNE by gesture for User 1



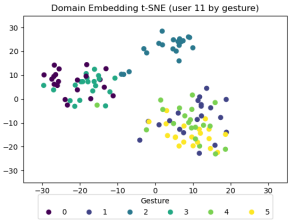
(h) t-SNE by gesture for User 3



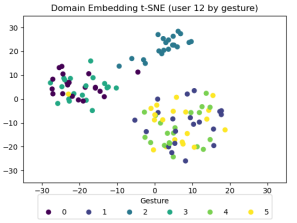
(i) t-SNE by gesture for User 5



(j) t-SNE by gesture for User 10



(k) t-SNE by gesture for User 11



(l) t-SNE by gesture for User 12

Figure B.2: t-SNEs of the latent space and domain embeddings produced by PPO with one-hot encoding and MTF transformation with extended training time, filtered by User.