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
Dr. Maryam Tavakol

Fourth Draft

Eindhoven, September 2023

Contents

Chapter 1

Introduction

In this ever more connected climate that we find ourselves in, IoT devices everywhere are adding little conveniences to our every day lives. With IoT devices becoming ever more common and reaching a forecasted 27 billion devices by 2025 [12], the dream of ubiquitous computing and sensing is transitioning from a mere dream to the reality of our every day lives. Additionally, approximately 19% of new devices bought in 2020 also utilize some form of Wi-Fi radio for communications with a forecasted increase to 24% by 2025.

It is clear that this trend towards integrating computing technology into everyday objects 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 becoming and edge computing ability comes ubiquitous sensing, enabling new modalities of interaction and improving data collection and ana-

Figure 1.1: A WiZ A60 Tunable White light bulb, one example of an IoT smart device with an integrated Wi-Fi Radio [30].

lytics. It is 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 call 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 does not require input devices. It is not too common to use the end device itself as the input. 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. 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.

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 Phillips Hue Light bulb in Figure 1.1, and it would be a rather safe assumption to make 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 of finer-grain signals to be extracted 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 is used.

With this thesis, we aim to explore the use of CNN 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 table-to-image and 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 CNN classifier 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) and this is at least partially credited as a reason the building has the best efficiency of any academic building in Europe when it was constructed [9].

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.

Finally, the deployment of 5G networks in densely populated areas is working towards enabling faster speeds and lower latency, essential in many ubiquitous computing and sensing applications.

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 [7]. 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 [6], the answer is a resounding yes!

There are even commercial products which have become available on the commercial market during the writing of this very thesis that incorporates parts 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 [29]. It should be noted, though, that this is only capable of presence detection and not of gesture recognition.

However, ML suffers from degraded performance when faced with domain-shifts. When dealing with new, unseen users and environments, gesture classification accuracy degrades significantly [6, 1, 24, 2, 13, 15, 42, 14, 19]. 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. Domain-agnostic systems are required for many industry applications, such as for patient sensing in healthcare, for consumer behavior tracking in retail, and for hands-free control of smart IoT devices in consumer homes. 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 [6, 1, 24, 2, 13, 15, 42, 14, 19]. It is thus interesting to be able to create a domain-agnostic 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 [33]. Various OpenAI projects, on the other hand, simply scrape large amounts of websites and collect these as part of their dataset [5]. Within the scope of our research, due to its nature, such widespread data collection methods are non-viable, and we must resort to more novel approaches towards data-agnosticism.

Most alternative methods to simply using very large datasets rely on a ground truth domain label being provided [15, 39]. In 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 maintaining enough information that a classifier model can use its output to correctly classify target features.

In [19], 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 [19] 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 CNN to accurately produce a latent representation of the domain space, measured by performance metric difference in a domain factor leave-out cross validation setting between a classifier provided the domain space representation and one which has not?

2. To what extent would changing the latent representation of the domain space 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 space representation types?

3. To what extent can signal-to-image transformation replace the state machine neural network presented in [19], measured by comparing the performance metric difference in a domain factor leave-out cross validation setting between the model presented in [19] and our approach provided no self-label?

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 [6]. 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 identification [24]. 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 [7]. This group is focused on standardizing the hardware requirements, specifically enabling CSI accessibility and specific measurement procedures which future devices can 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. [13]. 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 [36]. 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 [14]. 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 known subjects and ≈8–10% lower performance on new domains.

The same group, in the same year, published DeepMV which proposes the use of multiple APs and audio sources, in the form of ultrasound signals, with a domain discriminator and embedding generator [39]. This work is based on the intuition that the fusion of sensor data from multiple APs and audio sources placed around the room, which intuitively should provide more data. The embedding generator produces a latent representation of the action which can then be used by a fully connected module 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.

The 2021 paper by Zhuravchak et al. proposes the use of an LSTM as a classifier [43]. In this method 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 [19]. 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' fromt he 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 [4]. Their result shows $\sim$88% mean accuracy on Widar 3.0, dropping by $\sim$4% on out-of-domain test samples.

The 2022 thesis by Oerlemans compares how different preprocessing methods appear to affect gesture recognition performance [23]. 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 the Widar dataset for brevity, presents a dataset with developed specifically for "cross-domain learning solutions" [42]. 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 [20]. 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 $\sim$87%

mean accuracy over 150 sign gestures.

Person-in-WiFi is the dataset used in the paper by Wang et al. [36]. 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 tabular data into images for deep learning. Four state-of-the-art approaches are Gramian Angular Fields (GAF), Markov Transition Fields (MTF) [37], and Recurrent Plots (RP) [8]. 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 [26]. 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) [44]. 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 [40]. 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 [31]. 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.

Zhang et al. in 2021 published research into using RL based feature selection for domain shift mitigation [41]. 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 [21]. 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 label predictor loss, using Cross Entropy Loss, and the domain classifier loss, also 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 difference between two domains. The MMD measures the kernel-based difference between feature means of each domain. This work proposes the use of MMD with the goal of minimizing this distance while maintaining the distinctness of each domain, allowing for the domain classifier to still work.

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 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

Variational autoencoders (VAEs) are a type of generative model which combines elements of both autoencoders and probabilistic modeling and are suited for unsupervised learning and representation learning and was first introduced by Kingma and Weling in [16]. 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 [17].

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 [17].

By using the reparameterization trick introduced in [16], 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 [32]. 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. [27]. It is a further development based on Trust Region Policy Optimization (TRPO) [28] 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" [27].

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, where

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

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

The clipped surrogate object approach shows better empirical performance in [27] 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 is a model-free off-policy RL alrogithm and is an extension of Q-Learning which replaces the crafted Q-tables with neural networks [38, 22].

The concept behind DQNs is to use a deep neural network to approximate the optimal action-value function

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

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^- are the target network parameters. θ_i^- are only updated with the Q-network parameters θ_i every C steps and are otherwise kept fixed.

3.2.3 Deep Deterministic Policy Gradient

DDPG builds on DQNs and extends the action space to continuous action spaces [18]. 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

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

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

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

$$\text{let } a_i = \mu_{\theta_\mu}(s_i) \tag{3.11}$$

$$\nabla_{\theta_\mu} J \approx \frac{1}{N} \sum_i \nabla_{a_i} Q_{\theta_Q}(s_i, a_i) \nabla_{\theta_\mu} a \tag{3.12}$$

    ;

    Update the target networks:

$$\theta_{Q'} \leftarrow \tau\theta_Q + (1-\tau)\theta_{Q'} \tag{3.13}$$

$$\theta_{\mu'} \leftarrow \tau\theta_\mu + (1-\tau)\theta_{\mu'} \tag{3.14}$$

    ;

  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 $z = a + bi = re^{i\varphi}$ 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.

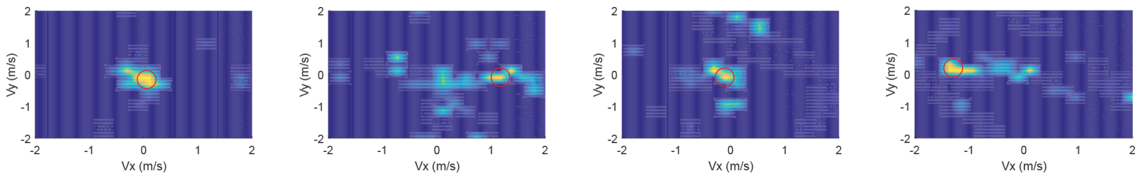
3.4 Body-coordinate Velocity Profile

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 and the intensity of solar radiation on a given day. To remedy this, Zheng et al. [42] proposes the use of the Body-coordinate Velocity Profile (BVP), a representation of the CSI which is theoretically independent of domain factors [42]. 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 [42].

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 [42] 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 derivation, which are somewhat unnecessary to the understanding of the rest of this thesis, can be found in Section 4 of [42].

Suffice to say, the BVP representation of a gesture is an $N \times N \times T$ tensor, where T is the



(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 [42].

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.

3.5.1 Gramian Angular Fields

GAFs [37] 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

$$\begin{cases} \phi = \arccos(\tilde{x}_i), & \tilde{x}_i \in \tilde{X} \\ r = \frac{t_i}{N}, & t_i \in \mathbb{N} \end{cases} \tag{3.15}$$

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

$$G = \left\{ \begin{matrix} \cos(\phi_1 + \phi_1) & \cdots & \cos(\phi_1 + \phi_n) \\ \cos(\phi_2 + \phi_1) & \cdots & \cos(\phi_2 + \phi_n) \\ \vdots & \ddots & \vdots \\ \cos(\phi_n + \phi_1) & \cdots & \cos(\phi_n + \phi_n) \end{matrix} \right\} = \tilde{X}' \cdot \tilde{X} - \sqrt{I - \tilde{X}^2}' \cdot \sqrt{I - \tilde{X}^2} \tag{3.16}$$

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

$$G = \begin{Bmatrix} <\tilde{x}_1, \tilde{x}_1> & \cdots & <\tilde{x}_1, \tilde{x}_n> \\ <\tilde{x}_2, \tilde{x}_1> & \cdots & <\tilde{x}_2, \tilde{x}_n> \\ \vdots & \ddots & \vdots \\ <\tilde{x}_n \tilde{x}_1> & \cdots & <\tilde{x}_n, \tilde{x}_n> \end{Bmatrix} \tag{3.17}$$

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 [37] 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

$$M = \begin{Bmatrix} w_{kl|x_1 \in q_k, x_1 \in q_l} & \cdots & w_{kl|x_1 \in q_k, x_n \in q_l} \\ w_{kl|x_2 \in q_k, x_1 \in q_l} & \cdots & w_{kl|x_2 \in q_k, x_n \in q_l} \\ \vdots & \ddots & \vdots \\ w_{kl|x_n \in q_k, x_1 \in q_l} & \cdots & w_{kl|x_n \in q_k, x_n \in q_l} \end{Bmatrix} \tag{3.18}$$

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 [8] 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

$$\vec{x}_i = \left(x_i, x_{i+\tau}, \ldots, x_{i+(m-1)\tau}\right), \ \forall i \ in\{1, \ldots, n-(m-1)\tau\} \tag{3.19}$$

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

$$R_{i,j} = \Theta\left(\epsilon - ||\vec{x}_i - \vec{x}_j||\right), \ \forall i, j \in \{1, \ldots, n-(m-1)\tau\} \tag{3.20}$$

where ϵ is a threshold and Θ is the Heaviside function

$$\Theta(x) := \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{3.21}$$

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.

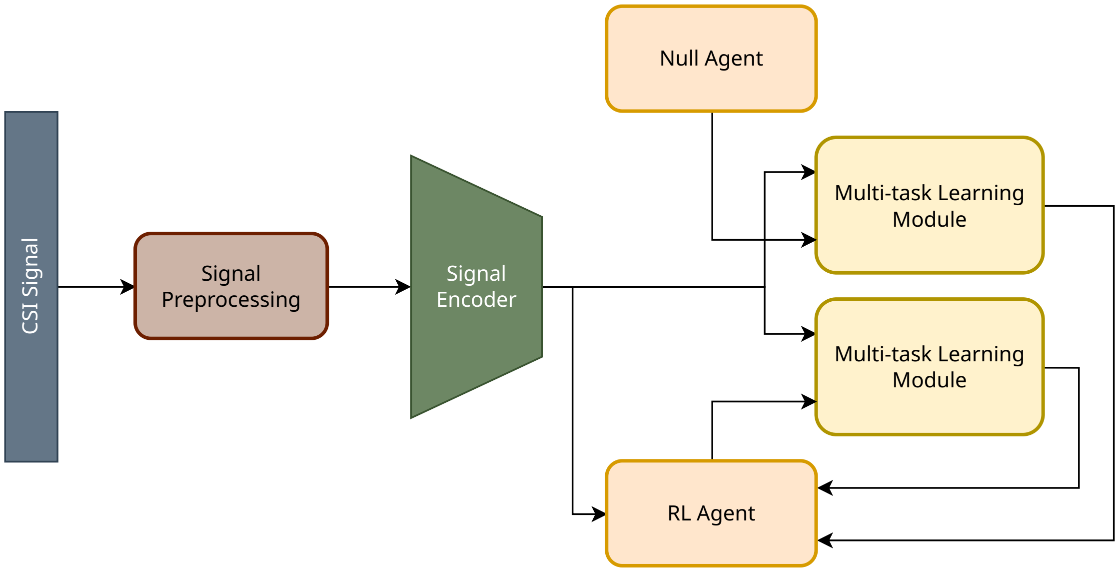


Figure 4.1: An abstracted diagram of the proposed DARLInG architecture.

4.1 Widar 3.0

The Widar 3.0 dataset [42] 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 8 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 [42].

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 4.1.

Selection Parameters	Training	Test
User	3, 5, 6, 10, 11, 12	1
Room Number	1	
Gesture	all	

Table 4.1: Selection parameters for samples in the training and testing set for single user leave out.

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-agnostic gesture identification using RL. Our intuition comes from [41] 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 VAE-style 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 preprocessing and signal-to-image transformation pipeline, described in Section 4.3 and 4.3.2. 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 of that environment at time t, s_t. An agent then performs an action a_t and 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 4.2.

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

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

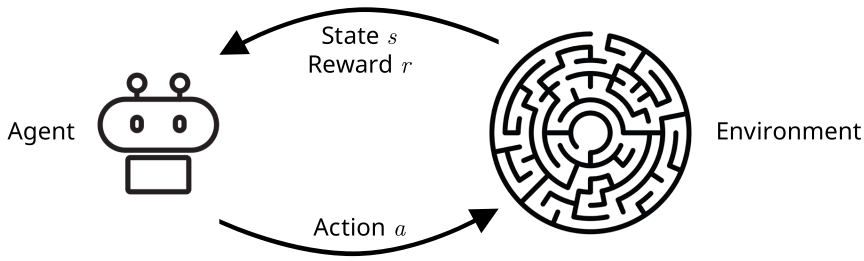


Figure 4.2: The basic paradigm of Reinforcement Learning where an agent interacts with its environment through actions a and receives a new state s and reward r in return.

dimensionality of the latent representation.

To provide the RL agent with a reward function, inspired by triplet loss, we use two different multi-task learning modules. The details of the reward function are described in Subsection 4.6.1. One of these modules is provided $\boldsymbol{d}_{\emptyset}$, representing a vector of zeros when the d is one-hot encoded or a value of $\frac{1}{f}$ when $\boldsymbol{d}_r$ is a probability measure, and the other $\boldsymbol{d}_r$.

4.3 Signal Preprocessing

As the old adage goes, garbage in, garbage out. As such, we propose a signal preprocessing module, visualized in Figure 4.3. This module first cleans the CSI signal using traditional signal processing and then transforms it into an image for use by the signal encoder.

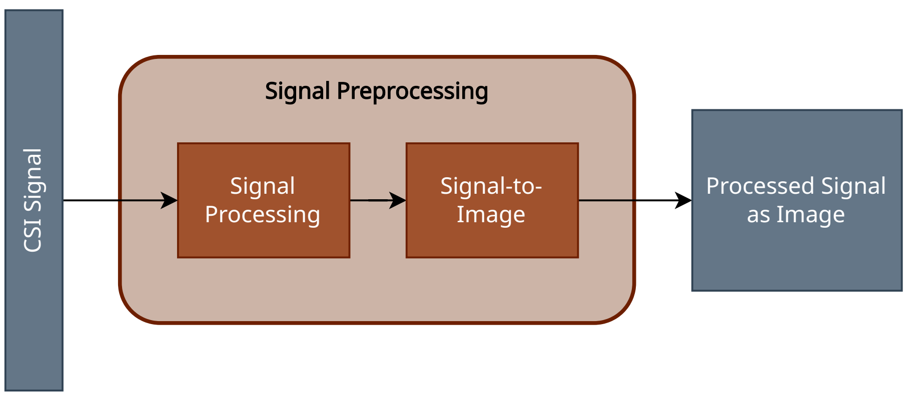
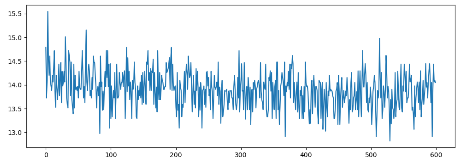
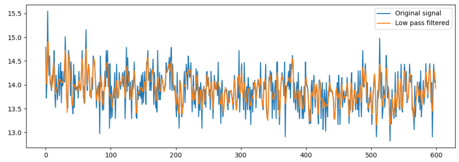


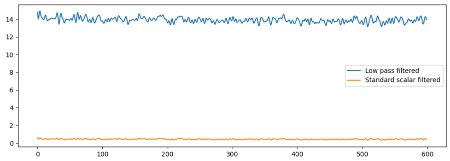
Figure 4.3: Details of the signal preprocessing module. The module is comprised of traditional signal preprocessing and a signal-to-image transformation.



(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 4.4: 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.

4.3.1 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 or its amplitude and phase shifts, 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:

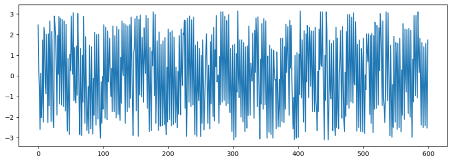
1. **Low Pass Filter** set to a cutoff frequency of 250 Hz and an order of 4.

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.

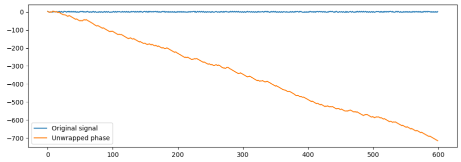
The low pass filter is used to eliminate noise inherent in CSI data from environmental factors. It has also been set to these values based on empirical experimentation which shows that no human movements during the performance of the 6 gestures which we test for has a frequency above 250 Hz.

The standard scalar is used since empirical evidence shows that neural networks work best when input values are close to the interval $[-1, 1]$ [11].

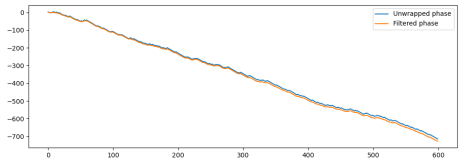
The effect of each step of the amplitude pipeline can be seen in Figure 4.4.



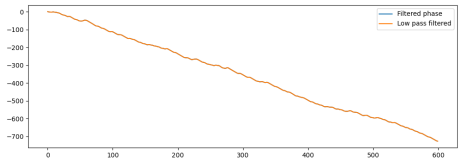
(a) The original phase signal.



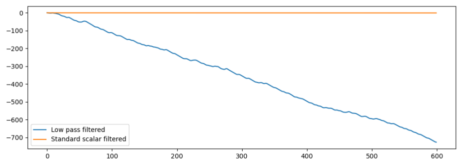
(b) The original phase signal and the phase unwrapped signal.



(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 4.5: 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.

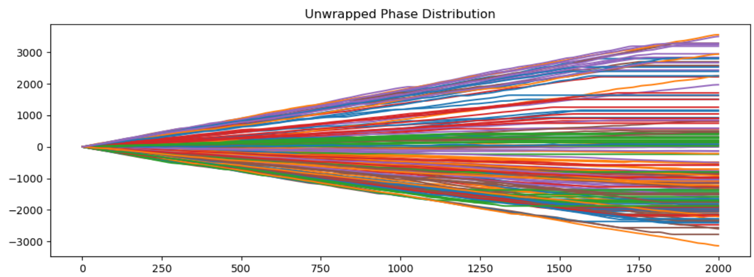


Figure 4.6: A plot over time of the phase shift after unwrapping of all training data, colored by gesture, showing how there is no general trend of only increasing or decreasing phases.

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, inspired by [23].

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

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 4.5 after phase unwrapping. Further investigation showed that this is actually 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 4.6. The plot clearly shows that there is no general trend of phases increasing or decreasing infinitely, with many phase shifts staying around 0 after unwrapping.

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. There are many works from the literature which show that there are certainly improvements that can be gained from performing image processing on temporal data. We will experiment with the following four methods for signal-to-image transformation: Gramian Angular Fields (GAF) [37], Markov Transition Fields (MTF) [37], and Recurrent Plots (RP) [8]. 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 [10], 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 double that of the provided BVP data in Widar 3.0. 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 4.7. We can then proceed with the encoding of this image into a latent space.

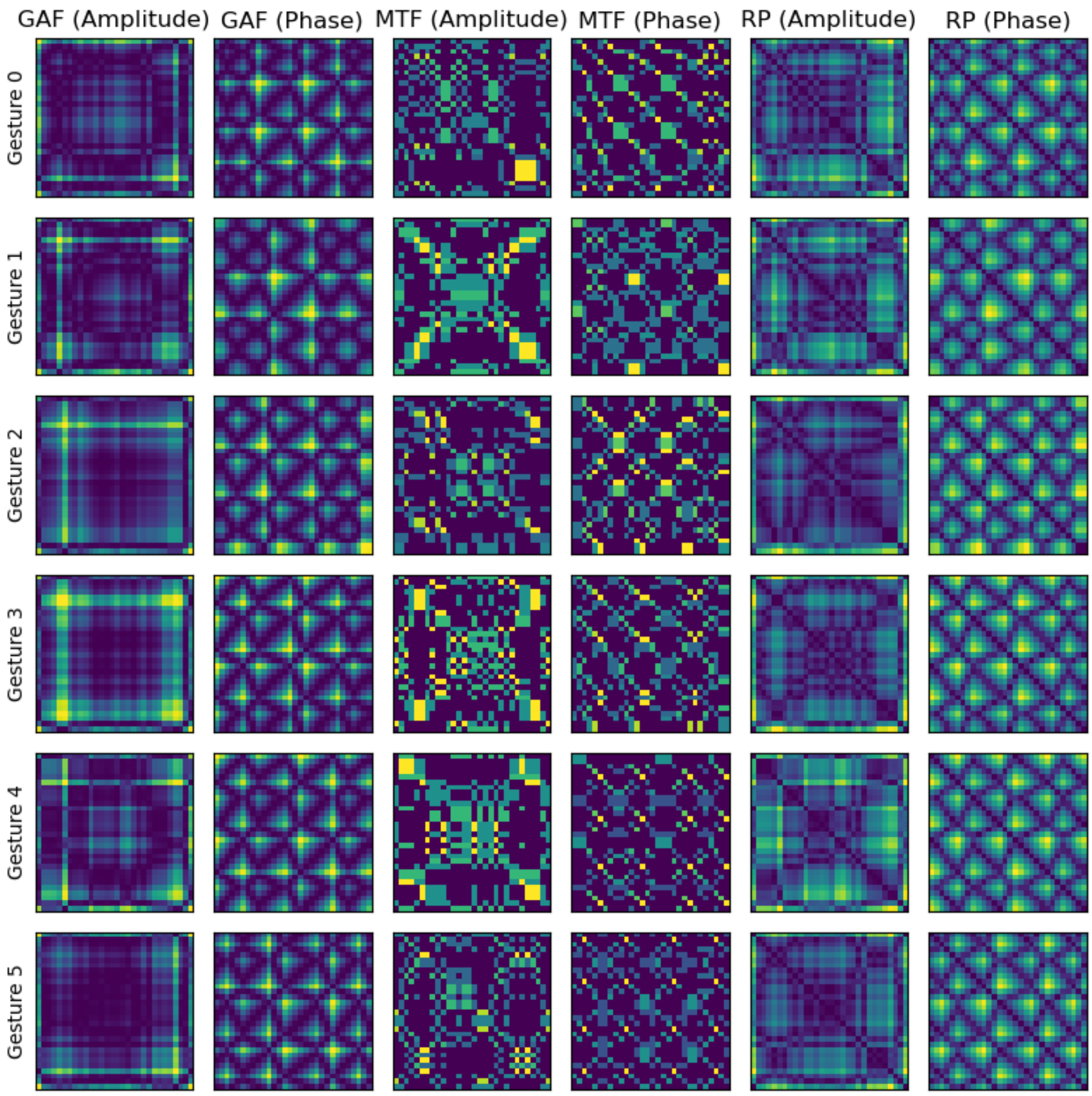


Figure 4.7: Samples of each gesture from the Widar 3.0 dataset being transformed using each of the three signal-to-image transformations. Each pair of columns is the transformation by a single signal-to-image transformations on the amplitude and phase signals, respectively.

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 `Conv2d` layer and its dropout is set dynamically and differs by experiment. The activation layer 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 of 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 z. A visualization of the CNN-based VAE used can be seen in Figure 4.8.

The latent representation 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. 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.

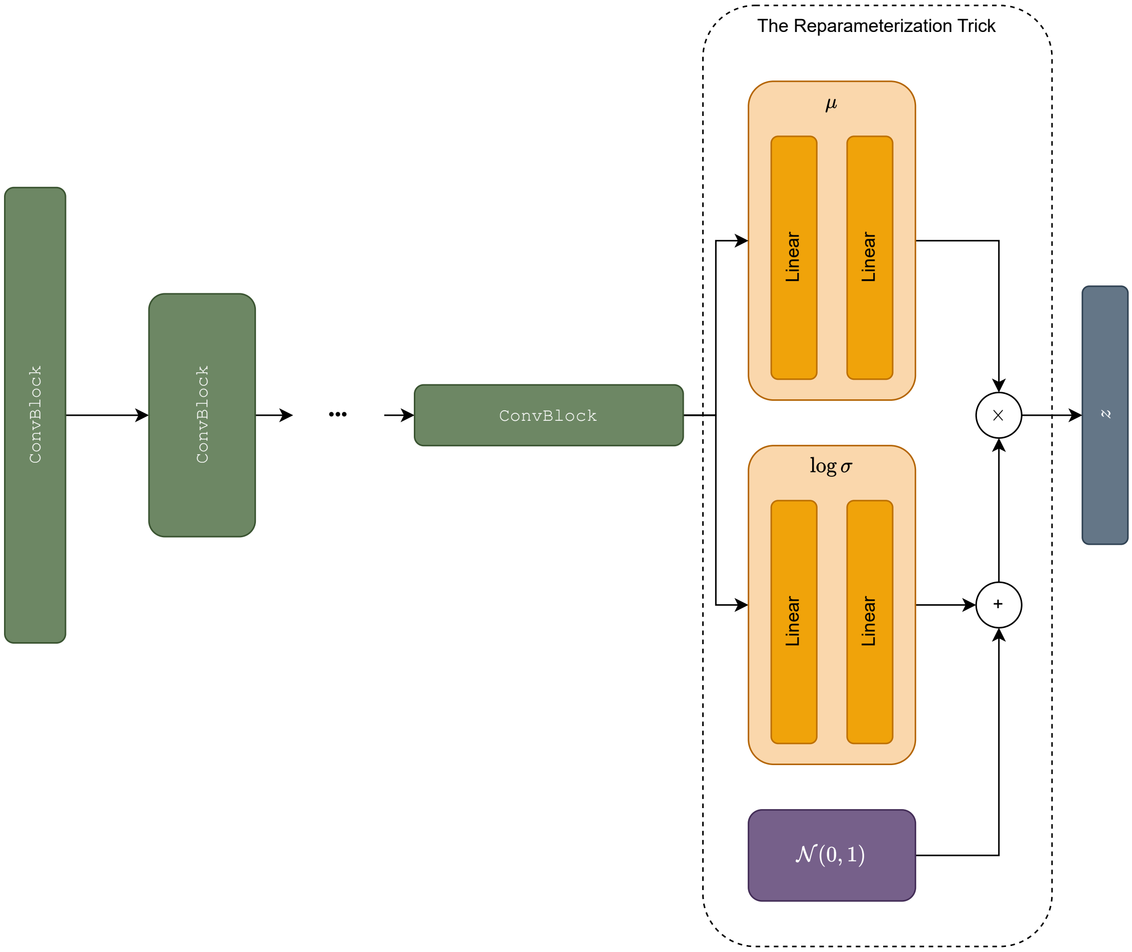


Figure 4.8: The CNN-based VAE used in DARLInG. The number of ConvBlock elements is variable, depending on the experiment being performed. The reparameterization trick is the same trick described in [16].

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 4.9. 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 [42], even though we are ultimately not interested in the BVP. This approach is inspired by Martini et al. [21], 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 [35]. 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]. Further, [21] supports giving the latent representation domain-discriminatory powers.

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. This comparison of the performance of the two embed heads is inspired by triplet loss.

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

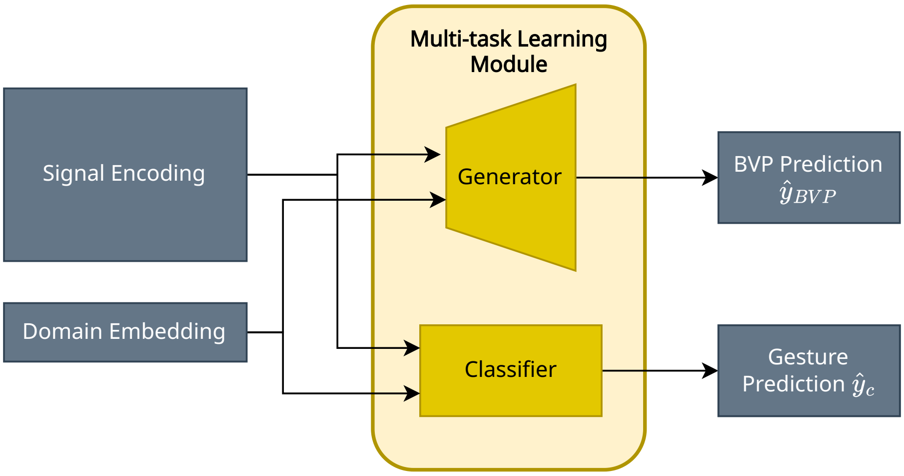


Figure 4.9: Details of the multi-task learning module. The generator is made up of a series of deconvolutional layers and produces the BVP prediction while the classifier is a fully connected network and produces a probability distribution over each gesture class.

4.6 Reinforcement Learning Agent

To mitigate domain shift, we implement a novel method for unsupervised domain representations through reinforcement learning. Recall the common terminology used in RL, the agent, the action a, the state observation s, and 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 4.6.1.

In this work, we will be looking at using both an on-policy and off-policy algorithm, PPO and DDPG, 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 [25], 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 [34].

Our use of RL to tackle this problem is inspired by both [19] and [41]. In [19], 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, [41] 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 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 ϕ 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

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

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 ϕ. 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. Additionally, 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 $\gamma \cdot \phi\left(\hat{y}_{c,\emptyset}\right)$

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 can be described as

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

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

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

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
        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;
    if k ≥ ζ then
        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;
```

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

[5] 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`.

[6] 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.

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

[8] 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.

[9] 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/`.

[10] 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`.

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

[12] 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).

[13] 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.

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

[15] 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.

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

[17] 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`.

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

[19] 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.

[20] 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.

[21] 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`.

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

[23] 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.

[24] 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.

[25] 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`.

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

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

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

[29] 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).

[30] 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).

[31] 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.

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

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

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

[35] 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.

[36] 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.

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

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

[39] 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.

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

[41] 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.

[42] 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.

[43] 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.

[44] 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.

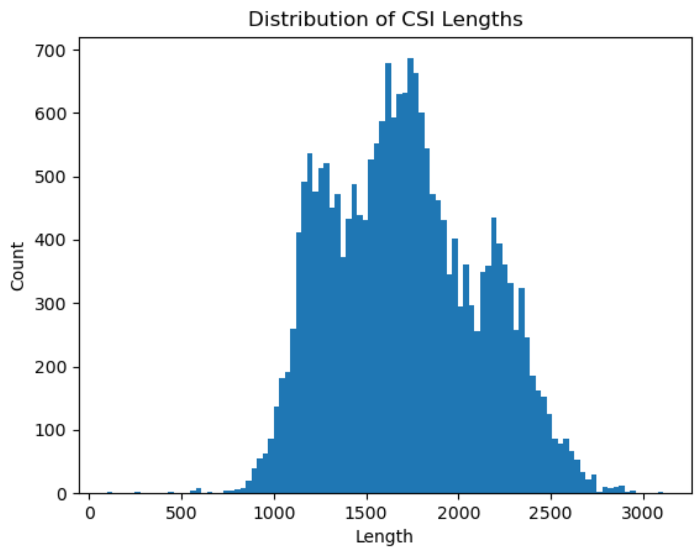


Figure A.1: Histogram of the length of CSI datapoints. CSI datapoints collected were of different lengths with the majority being less than 2000 samples long, representing samples of around 2 seconds each.

	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.