BACHELOR

Insights on Robustness of Adversarial Deep Learning Model for WiFi-CSI Data Domain Factor Independent Feature Extraction

Sha, Tsatsral Mendsuren

Award date:
2022

EINDHOVEN UNIVERSITY OF TECHNOLOGY

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

Insights on Robustness of Adversarial Deep Learning Model for WiFi-CSI Data Domain Factor Independent Feature Extraction

Bachelor End Project Report

Tsatsral Mendsuren Sha

Supervisors:
Nirvana Meratnia
Bram van Berlo

27-06-2022

Contents

Chapter 1

Introduction

1.1 Context and Topic

Sensing techniques, when embedded in intelligent engineering designs, have broadened the functionality of countless successful human-machine interaction applications to help humankind more automatically and intelligently. The realization of such applications involves developing sensing techniques and hardware and the subsequent understanding of the sensed information so that the hardware may react adequately. For example, self-driving cars, which can sense objects on the road and react accordingly, and the next-level human-computer interaction employing gesture recognition instead of touching the screen, would not have existed without numerous research on the interpretation of the sensed data.

Among all applications involving sensing techniques, the contact-free interactions between humans and systems are one of the most appealing to realize. As one can imagine, this may free users from relying on the hardware devices for interaction, expanding its target users to people with physical difficulties and people with less experience using digital devices. Without the participation of traditional hardware devices, how to teach the systems to understand users' instructions correctly becomes the first problem to solve. Luckily, Wireless Fidelity (WiFi) signals, which are widely present nowadays due to the increasing need for high-quality information transmission, can be utilized to sense human activities. What is unique about it is that human activities can influence WiFi signals and the Channel State Information (CSI); therefore, human activity information can be embedded in the WiFi-CSI data and extracted when this data is well interpreted.

More specifically, CSI includes many indicators, such as amplitude, phase, and signal delay of the carrier signals. With the changes in the transmission

distance, these indicators could reflect the corresponding signal changes that occur with the carrier [1]. Therefore, when studied adequately, the changes in CSI data can be conversely interpreted to study the changes in the physical environment of the WiFi communication, in other words, to realize device-free sensing [16]. Furthermore, due to its high sensitivity to environmental changes, one positive consequence is that the apparent movements of humans and objects are reflected by it, and the minor and subtle movements, such as hand gestures, can be captured and analyzed. However, at the same time, many other minor environmental changes that we are not interested in may also embed in the WiFi-CSI, which complicates our analysis.

One state-of-the-art method is deep learning to extract human activity information from the WiFi-CSI data. The basic idea of deep learning is to concatenate submodels as layers where each layer has many computation nodes and then combine every layer with an activation function. Training data is then iterated through this structure, and essentially a mapping function from the data to, for example, a classification task, could be derived. This mapping function could be well optimized and achieve high performance in highly complex cases where no simple linear relations can be derived. Some level of automated computation might be needed [6]. In our case, it is applied to analyzing the WiFi-CSI data.

In recent years, the CSI data captured by WiFi devices has been widely used for human activity recognition utilizing deep learning. The underlying logic is that when a person is moving or making different gestures between the transmitter and the receiver of the WiFi signals, the CSI of the WiFi signals will vary accordingly. It follows that specific patterns of the CSI could be associated with the specific activities that may cause these patterns. The context and the topic of this research will focus on applying the deep learning model on human activity recognition with WiFi-CSI data. More details will follow in the rest sections.

1.2 Problem Statement

Ideally, the CSI is only affected by the human activities we would like to study. However, WiFi-CSI data can also be affected by many known and unknown factors, such as the device's location and orientation. The signals could also behave differently even when the human activities are essentially the same but done by different people potentially with different physical properties [33]. These factors are often addressed as "domain factors."

The presence of domain factors in the environment may cause a shift in the distributions of the training data and the inferring data, which then affects

the generalization of the deep learning model. To achieve high robustness against the possible domain shift, the algorithms must adapt well to different scenarios, possibly with different domain factors. In other words, when testing the algorithm with data influenced by unseen or unknown domain factors, we should not see a significant drop in the algorithm's performance. However, the problem, in reality, is that the optimization of the model is often not free from domain factors' effect; thus, the influence of domain shift can reduce the inference performance of the model.

1.3 Research Motivation

In recent years, many solutions have been proposed to address the domain shift issues mentioned above. The most straightforward solution is to increase the size and variations of the training data so that the classification model could be exposed to as many environments as possible; for example, large tech companies like Tesla could do this as they have access to an enormous amount of data [4]. However, substantial storage, training time, and computational power are needed for a large amount of training data, which is not always feasible for all research projects. Another direct solution might be manually creating features for training that are not influenced by domain factors. However, this method is difficult and highly unreliable since such creation requires domain knowledge in certain fields and may introduce manual biases [7].

Another promising research option is to extract domain-invariant features from the deep learning model. One way to do this is through adversarial training. A component called "discriminator" is implemented besides the structure for the main task (for example, human activity recognition). The discriminator will make predictions for the domain factors. A min-max game on loss functions is played so that the goal of training is to extract some commonness in the features independent from domain factors. Plenty of evidence showed that architectures based on this idea can achieve good performances when applied to test data with domain shift [17][24][26][32], both for general image classification suffering from the influence of domain factors, and for applications of WiFi-CSI data for human activity recognition. Moreover, authors of [21] implemented an adversarial model using newer neural network components, which achieved high performance in the domain leave-out-cross-validation setting.

However, such adversarial training depends on the availability and correctness of the domain labels since the discriminator's training on the domain factor classification is supervised. Thus, the model might not be as robust

whenever new domain factors have to be considered; thus, have to be re-trained from scratch with new data. Moreover, labeling all the potential domain factors is not only manually intensive but also technically unrealistic. There are always unknown domain factors present even in the strict lab environment far from the noisy real-world environment. At the same time, we cannot ignore that the adversarial models still achieve the best performance against domain shift in their traditional setting; therefore, adversarial models could serve as the baseline where we seek more improvement.

Therefore, for the adversarial model, to mitigate the above-mentioned main constraint - that it still relies on the availability of the domain labels and requires re-training from scratch for new domain factors - an overall research motivation is to further minimize the impact of domain factors on the adversarial model. For example, the model in [21] can be used as a baseline model for investigation.

Another method - reinforcement learning - may inspire further improvement in mitigating the domain shift issue in human activity recognition with WiFi-CSI data. As the basic components of a reinforcement learning agent and its optimization process do not depend on labels, the reinforcement learning agent can potentially be used to deal with the dependency of domain labels in adversarial learning. However, very few methods regarding reinforcement learning have been investigated on WiFi-CSI data. A very recent research [15] introduced a reinforcement learning agent to train an RNN component used for optimizing the neural network architecture for human activity recognition. In their research, a domain factor invariant latent representation can be extracted without the help of domain labels. This research on the reinforcement learning component inspires a future research direction since this component can be utilized without the limits imposed by the availability of the domain information.

In order to mitigate the above-mentioned main constraint, a straightforward intuition is to mix as many domains as possible in training. However, we are still restricted by the availability of the annotated domain labels. Therefore, the possibility of using an automatic method to correctly label the domain factors while achieving high inference performance and domain-independent feature representation could be investigated. For example, realizing this by a reinforcement learning agent. More specifically, when the real domains factors present in the data are unknown, this agent should be able to label the domains of the data, for example, to properly change the combination of domain factors that could be present in the data and also to decide the domains to be accounted for in the output of the discriminator.

However, another problem with such an auto-labeling procedure is that it brings new domain information to the training, as it tries out possible

combinations of domain factors to optimize the model (in which some domain factors might not have been seen by the model yet). The effect of mixing new domain factors in the previously domain-specific training cannot be derived directly. Most importantly, if the effect of mixing new and messy auto-labeled domain factors will reduce the extent of maintaining a good domain-independent feature representation during the training, we will need to properly define, for example, a reward function related to the intermediate representation for the agent to prevent this. Hence, what is left unclear is how we should define such a reward metric to teach the agent to label the domain factors properly so that the intermediate representation can remain domain-invariant as much as possible.

Therefore, the first step for this approach is to get the information on whether and how the model reacts to the presence of unseen domain labels that the reinforcement learning agent may make. This step is necessary as it may give insights on how to define heuristics in the reward function or help to know whether the current model can be extended with such a reinforcement learning agent. In this thesis, this step is our main focus.

For this first step, one method is to manually "imitate" the possible actions of the agent in a known and controllable way and investigate the corresponding results. For example, insert a known amount of false domain labels - labels that are different from their true label - in the training data and investigate the robustness of the model under such environments. Hence, a more precise goal of this research is defined. That is to perform the necessary experiments of introducing false domain labels to understand the agent's possible effect on the model. For example, the effect on the inference performance and also on the latent representation of the model. In this regard, the main research question to achieve this goal is:

- To what extent is the model robust against domain mislabels in the domain specific learning phase?

In order to define a more specific experiment setting for this thesis, two sub-research questions are defined:

- To what extent will the domain-independent feature extraction change when the different amounts of false domain labels are present during the domain-specific training phase?
- When there is domain shift from the domains used for training, to what extent will the domain-independent latent representations change when the different amounts of false domain labels were present in training?

1.4 Findings

Unfortunately, this thesis cannot provide conclusive results on the robustness of the model against domain shift. The model did not sensitively react to the several experiments we performed. We further discussed the possible reason for this, including the issues with the data, the experiments and the model architectures. Hence, this research may provide a basis for discussion future research continuation. Necessary future work has to be done addressing the limitation of this research.

Chapter 2

Background

In recent years, many approaches have been applied to achieve domain adaptation in broader contexts with domain shift issues. In order to have a complete understanding of the prior works to this thesis, we reviewed methods belonging to the field of deep learning, including supervised learning, unsupervised learning, and reinforcement learning. Methods reviewed are not exclusive to human activity recognition using WiFi-CSI data, so the insights from other research fields, for example, image classification, can also be included as research inspirations.

2.1 Supervised Learning

Supervised learning aiming at domain adaptation, just as its name indicates, often requires the availability of specific information regarding the domain factors in the data; this information is usually annotated as domain labels next to the task labels (or target labels), which describe the task, for example, human activities, that we are interested. For example, one prior work proposed a deep learning model called SignFi [17] based on convolutions neural networks for hand sign recognition. Both the target labels (hand gestures) and domain labels (person and environment) are present. However, the performance is not robust when applying the model in shifted unseen domains. However, This paper points out that a common issue for the domain adaption problem is that the model that relies on the domain labels does not often generalize well whenever a new domain is seen.

Some research experimented with generating virtual features for target domains so that the model also accounts for the target domains. One example is WiAG [23]. A translation function between the domains was adopted to generate virtual data for possible domain environments, based on a high-level

understanding of the knowledge in sensing with WiFi-CSI data. CrossSense [29] adopted a method to learn the correlations between features to translate unseen domains into new virtual feature space properly. WiSign [30] generated handcrafted features during pre-processing of the WiFi-CSI signals; the generation is done by defining a cross-correlation function that replies on the domain-specific information and knowledge. Nevertheless, methods in this direction would have scalability issues and always require knowledge in the physical field regarding the effect of domain factors on the signals, thus introducing label bias [7].

Instead of generating handcrafted feature spaces, other solutions based their methods on altering the deep learning models. For example, many solutions focused on extracting a domain-independent feature in the environment of latent representation. This idea often requires the steering of the model utilizing the domain labels during the training in order for the model to learn the common features of the data instead of the features differentiating the domain information. One way to realize this is through the adversarial network, which often includes a part aiming at mapping the domain environments. The loss of this component is often maximized in order to force the task recognizer only to learn regardless of domain factors. For example, [32] uses an effective conditional adversarial architecture to maintain the common representation for the classification task on radio signal data. For WiFi-based gesture/human activity recognition applications, [24], [26] and [14] all adopted models based on adversarial training and have proved the effectiveness of this model in their experimental setting with domain shifts. However, Another problem with adversarial training on domain adaptation is that the discriminator always needs to train with available domain labels. Thus, the model's prediction may be less robust when the domain labels are unavailable or when facing unseen domain factors that are not included in the training. To include unlabeled data into the adversarial training, authors of [10] proposed the EI framework with several constraints and regularization methods embedded. However, the inference performance of their model is less satisfactory when facing domain shift.

2.2 Unsupervised Learning

In the previous sections, we have mentioned that the success of many prior works still relies on the availability of the domain labels, which might limit the generalization of these methods. To find the common representation across the domains without the help of domain labels, the authors of [33] introduced a method mainly focused on extracting the domain-independent signal from the raw WiFi-CSI dataset. This domain-independent signal – the gestures' velocity profile- is proven theoretically independent of the influence of domain factors. The experiments showed that their method has very robust high performance across domains. However, though proven to be effective, the velocity profile might only be informative in certain types of relatively simple gestures.

The more generalized idea is to incorporate unsupervised learning components in the deep learning models facing the problem of domain labels; for example, several methods have been tried to alter the adversarial training method mentioned in Section 2.1. The idea of adversarial learning is usually adopted to remove the embedded domain-dependent information in the latent representation of the features. This method is further investigated when data labels are missing, that is, in an unsupervised learning context. Such applications have been proven effective in other fields other than sensing with WiFi-CSI data. For example, the models in [5] are among the earliest experiments to add the domain discriminator trained in an unsupervised manner.

Following the idea of tolerating the unavailability of domain labels in adversarial training, many unsupervised solutions are proposed for sensing with WiFi-based signals. For example, in the [10], both labeled and unlabeled data was used to train the neural networks using the adversarial learning method. However, This method is still not ideally unsupervised since the discriminator always needs the information on the distributions of available domains to predict the unlabeled domain factors. Another example that utilizes unlabeled data is the model proposed in [34]. The model adopts unsupervised adversarial domain adaption after the classifier is well trained on the source domain. In the adversarial component, the encoders for both the target domain and source domain share a latent feature space, after which a domain discriminator forces the distance between the mapping of target and source to decrease by reducing the performance of the source domain label prediction. [11] considers multiple source domains instead of one single source domain in their training. Another research that is worth attention is [28]. Their proposed model uses pseudo-labeling and consistency regularization to align the target domains without the involvement of many model components

and training efforts, such as adversarial training. Although, in principle, all these methods do not need the collection of domain labels, re-training the classifier when applied in new environments is still needed as the features from the target domain play an important role in the training.

2.3 Reinforcement Learning

Further, the idea of reinforcement learning is also newly implemented in sensing with WiFI-based signals so that the model does not need to rely heavily on the domain labels. However, we noticed that some models usually stop generalizing well with the increase in the number of domains. This is because the model designs introduced in previous sections often more or less have to assume the domain factors' effect on the model's learning.

However, reinforcement learning naturally does not need such assumptions. Reinforcement learning is a training method to introduce an agent with actions that could steer the training directed by the agent's reward. In principle, reinforcement learning requires the definition of the rewarding/punishment towards certain behaviors, thus might be more functional in complex contexts when the manual model design or feature augmentation are hard to foresee without a large number of training efforts [20].

For example, [15] proposed a model using WiFi-CSI data for human activity classification facilitated by a reinforcement learning agent. Apart from the traditional deep learning components such as convolutional neural networks to predict the activities and improve the performance for activity recognition, a neural network architecture for optimizing the architecture of the activity recognizer is trained by reinforcement learning. As a result, the performance of the model increased not only on the test data from the domain for training but also maintained well on the unseen data with domain shifts. However, since the idea of reinforcement learning is newly introduced in human activity recognition using WiFi-CSI data, the amount of available research is still limited.

In conclusion, the experimentation with reinforcement learning agents points out a potential promising research direction in achieving domain adaptation with WiFi-CSI data, that is, implementing a proper reinforcement learning agent on a pre-trained architecture, for example, an architecture based on adversarial training. This method might not only hypothetically inherit the achievement of the prior work but also mitigate the above-mentioned limitations of supervised and unsupervised learning. Hence, the positioning of the research question concerning all the prior work serves as a start point for this research direction.

Chapter 3

Methodology

As we mentioned in the Section 1.3, the purpose of this research is to test how robust the domain-specific model is against the mislabels that are initially not included in its domain-specific training phase. In other words, for a deep learning model for domain-independent feature extraction trained with specific domain information, our experiments aim to provide information on this model once it no longer has access to fully correct domain information. Such a situation could happen when a reinforcement learning agent interacts with the model, such as facilitating the auto-labeling of the domain labels when domain labels are unknown. Hence, this thesis might also provide insights into the reinforcement learning agent to define its starting properties. Therefore, the experiments' principle is defined to fulfill this goal as much as possible. More detailed motivations and corresponding settings of the experiments are included in Section 3.2.

In the research thesis mentioned in Section 2, we have seen that some deep learning models for extracting the domain-independent features using adversarial training have already gained good inference performance except when domain factors are not clearly known in the target domain. Therefore, we notice that models of this structure could be used as the baseline model in which we seek advancement. In this thesis, we choose the adversarial model with the traditional domain discriminator by the authors of [21] to perform the experiments. The overview of this model is included in Section 3.1.

3.1 Model overview

Figure 3.1 shows the overview model we used for experiments. It is a model trained with adversarial training for domain-independent feature extraction. The following sub-sections introduce the components of this model that are

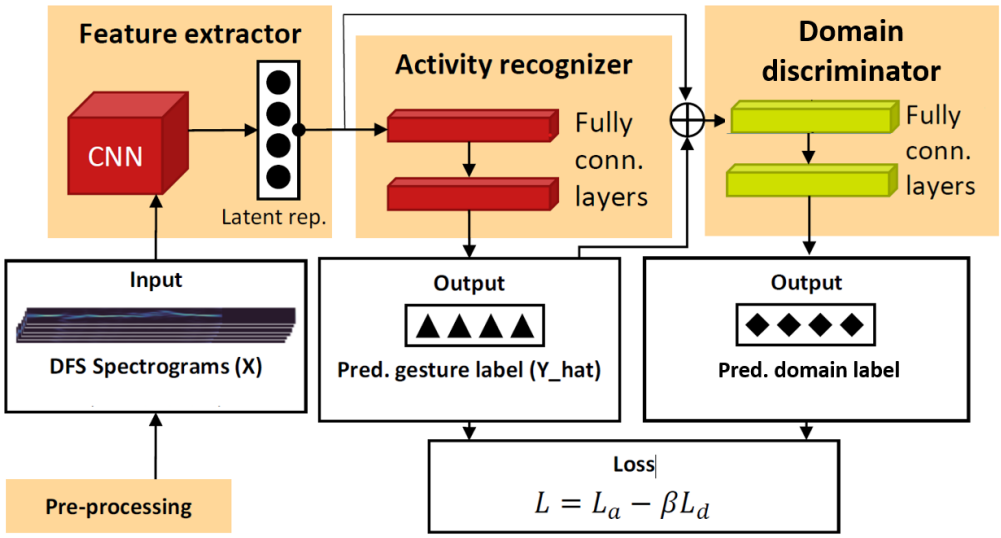

Figure 3.1: Pipeline of the model used for experiments [21]

relevant to our research.

3.1.1 Feature Extractor

The feature extractor is referred to as the backbone of this model. The purpose of this component is to extract domain-invariant features and create the latent representation of the features. In the main experiments of this research, we used the same backbone from [21] without altering its architecture and hyperparameters. The main part of this extractor composes two CNN layers with five mobile inverted bottleneck [18] and squeeze-an-excitation blocks [9] in between. Please see more details in [21].

3.1.2 Activity Recognizer

The purpose of this component is to predict gestures using the features passed by the feature extractor. It consists of two dense layers in which the last layer is applied with Softmax activation to output the probability distribution of the gestures to be predicted. We used the same activity recognizer from [21] as well.

3.1.3 Domain Discriminator

The output of this component is the prediction of domain labels. However, the output of this component is not of interest in performance evaluation as it only serves as a component to play the adversarial min-max game on the eventual loss. The domain discriminator consists of several fully connected feed-forward layers. In order to perform the adversarial training, the input to the domain discriminator is the output of the backbone concatenated with the gesture prediction from the activity recognizer. In our experiments, the number of the feed-forward layers is chosen from [2, 4, 6]. The layers are constructed in the following way. If the domain discriminator consists of 2 layers, the number of neurons is 248 and 150 for the two layers, respectively; if the depth of the domain discriminator is 4, the number of neurons in each layer is 460, 354, 248, 150; if the depth of the domain discriminator is 6, the number of neurons is 490, 420, 350, 280, 215, 150, respectively. These hyperparameters are manually determined so that the number of features after the backbone could gradually decrease to the number of domains. The last layer uses Softmax activation to produce the output, and the rest of the layers use ReLu activation.

3.1.4 Eventual Adversarial Loss

The model is trained by adversarial training; hence, an overall loss of the loss from the activity recognizer and the loss from the domain discriminator has to be defined and minimized. Such loss function is defined in equation 3.1, where β is a weight that equals 0.000001, L_a is the loss of the activity recognizer, and L_d is the loss of the domain discriminator. In this equation, β controls how much influence of the discriminator loss is added to the eventual loss such that the eventual loss can be minimized while the discriminator loss is maximized.

In the research thesis mentioned in Section 2, we have seen that some deep learning models for extracting the domain-independent features using adversarial training have already gained good inference performance except when domain factors are unknown in the target domain. Therefore, we notice that models of this structure could be used as the baseline model in which we seek advancement. In this thesis, we choose the adversarial model with the traditional domain discriminator by the authors of [21] to perform the experiments. The overview of this model is included in Section 3.1.

$$L = L_a - \beta L_d = -\frac{1}{|\hat{Y}|}\sum_{i=1}^{|\hat{Y}|}\sum_{j\in c_a} y_{i,j}\ln \hat{y_{i,j}} + \frac{\beta}{|\hat{S}|}\sum_{i=1}^{|\hat{S}|}\sum_{j\in c_d} s_{i,j}\ln \hat{s_{i,j}} \tag{3.1}$$

For other parameters in equation 3.1, $|\hat{Y}|$ and $|\hat{S}|$ is the batch size; c_d is the set of all the gestures, c_d is the set of all the domains; $y_{i,j}$ is the true label of gesture j in batch i (0 or 1), $s_{i,j}$ is the true label of domain j in batch i (0 or 1); $\hat{y_{i,j}}$ the predicted gesture value (0 or 1) for gesture j in batch i $\hat{s_{i,j}}$ the predicted domain label value (0 or 1) for domain j in batch i.

3.2 Experiments design

In Section 1.3, we have detailed stated the motivations of doing this research. To fulfill the research goal, in Section 1.3, more specific research questions were also defined to facilitate the design of the experiments:

- To what extent will the domain-independent feature extraction change when the different amounts of false domain labels are present during the domain-specific training phase?
- When there is domain shift from the domains used for training, to what extent will the domain-independent latent representations change when the different amounts of false domain labels were present in training?

Following the research questions, the overall setting of the experiments is to artificially introduce false domain labels in the training phase, which should have been exclusive to certain domains. We will also need to change the number of false domain labels to see whether and how the impact of such false domain labels will vary with respect to the amount of the false labels. The steps of the experiments we performed are illustrated in Section 3.2.2.

Moreover, we need to have tangible measurements to answer the sub-research questions, that is, to evaluate the exact impact of the mislabels. Specifically, for the first sub-research question, we choose the inference metric values on the validation data held out from the training data. The validation data is drawn theoretically from the same distribution as the training data but was not inserted with the false domain labels; therefore, inferring on the validation data could tell us how the model changes from the domain-specific space during the training. For the second sub-research question, we choose to observe the characteristics of the latent representation. More details of the evaluation strategy are included in Section 3.2.1.

3.2.1 Evaluation Strategy

The experiments were done in a leave-one-domain-factor-out cross-validation setting to evaluate the model's domain adaptation with domain shift. When one domain factor class (i.e., orientation) was left out in training and only used for testing, data was split with 40% as test data and 60% for training. The training set was further split by 10-fold cross-validation where the first fold with randomly selected samples was used as the validation data, and the rest folds formed the training data.

In order to answer the first sub-research question in Section 1.3, we reported the inference metrics on the validation data. During the training of

the above-mentioned model, based solely on a held-out validation subset, we measured and reported the classification performance metrics, including accuracy, precision, recall, f1 score, and Cohen Kappa across epochs and on the final model. Including multiple metrics instead of only the accuracy may help report more information when the dataset is skewed or imbalanced. The inference on validation data may help us see the effect of the mislabels on the model during the training as the validation data is essentially from the same environment as the training data but without the inference of false domain labels.

In order to answer the second sub-research question in Section 1.3, when training finished, we captured the latent intermediate representations after the backbone of the held-out test set. In order to visualize the representations, this latent representation was dimensionality reduced with t-SNE [22] and plotted on a two-dimensional plane.

The parameters for training are in line with the parameters used [21]. Training of the model uses Adaptive Moment Estimation (ADAM) [13] optimization with a learning rate of 0.0001. The number of epochs is 50, and the batch size is 12. Initially, the weight of the domain discriminator is set to 0.000001. Validation loss is used as a stopping criterion in early stopping used for training, where the minimum delta is 0 and patience is 5.

3.2.2 Steps of experiments

Inserting noise offline

The first step we carried out to investigate the impact of false domain labels is to directly alter the current domain label space of the data with false domain labels. In the beginning, since we do not yet know the effect of such false labels, a preliminary and straightforward way is to replace the true domain labels with a fixed amount of false ones. Inserting a fixed amount of domain label noise is a simplistic method to include unseen domain labels in the current domain-specific training. Yet, a method that greatly influences the training result since directly changing the data used for training will artificially confuse training with mislabels. Hence, the results from these experiments can tell us whether and how much the model is robust against the false domain information. In the meantime, the clean validation data is used to determine the stop point and evaluate the model's true performance in the training data's original environment. The inference results on the validation data could tell us to what extent the mislabels influence the training.

For comparison, different percentages of false domain labels were generated and inserted into the data. With this step, we could see how sensitive

the model is in reacting to the presence of false domain labels. We generated ten different domain label sets offline based on the domain labels of the dataset used for experiments. Each domain label set has a certain percentage of randomly chosen instances to be inserted with a randomized domain label. Such randomized false domain labels are referred to as noise in domain labels. Since the domain label of each observation is one-hot encoded, the noise was then created by inserting a one at a random index of an array made up of all zeroes. The percentages include 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 100%. All the random processes here return their value from a uniform distribution. These domain label sets replaced the true domain label sets in training.

However, one concern of the first method is that the noise positions might have uneven densities in the domain label sets. Thus, when a batch is drawn from the domain label set, percentage of the noise in each batch may not be ensured. For example, the overall percentage of noise in a domain label set is 20%; however, the amount of noise in each small batch could deviate from this value. We suspected that this was why we did not see many variations in the results of this experiment (see Section 4.2.1). Since the deficit of this method could cause the seeming robustness of the model, the second experiment was set up.

Inserting noise during training

The second experiment method serves the same purpose as the first experiment - to provide information during the model training when a different but fixed amount of false domain labels are in the training data. However, one different setting is that in this method, we insert the noise during the training instead of generating the noisy domain label sets offline. We refer to this second method as inserting noise online. Specifically, the replacement of true domain labels with the randomized false domain labels was realized during the training directly for every batch. This way, we can ensure the exact amount of mislabels fed to training as desired. Similar to approach one, the percentages of noise include: 10%, 30%, 50%, 70%, 90%. The weight and the size (number of feedforward layers) of the domain discriminator are altered to increase the reliability of the results,

However, as you can see in Section 4.2.2, the results we get from the second experiment are similar to inserting the noise offline. Therefore, we decided to further see whether this is due to the influence of domain labels in training through the domain discriminator is too small for the model to pick up any differences. A third experiment was set up.

Weight and depth of domain discriminator

In equation 3.1, we can see that β is the parameter that determines the weight of the loss from the domain discriminator (see Section 3.1.3) added to overall loss that determines the back-propagation during the training. If this value is too small, the influence of the information passed from the domain discriminator would not significantly influence the training.

Moreover, the domain discriminator is essentially a classifier in which the error is maximized instead of minimized. However, this does not mean that the architecture of the domain discriminator is not important. In fact, unlike using the information from correctly predicting the domain labels, deliberately wrongfully predicting the domain labels still requires the domain discriminator to correctly pick up the *informatively wrong* information of the domain factors for the back-propagation. Hence, another reason for the insensitive inference performance of the gesture prediction could be because the discriminator does not learn enough.

Therefore, the third experiment was to alter the weight and the number of feedforward layers in the domain discriminator to add more influence of the domain discriminator on the whole model.

Architecture of the feature extractor

Lastly, we also performed experiments replacing the feature extractor of the model using the standard model provided by Keras. The reason why we added this experiment is to rule out the fact that the model from [21] does not react to domain label noises is actually that the backbone of it generalizes too well given domain shifts. The chosen models were ResNet50 [8] and Xception [3], the less new neural network architecture compared with the feature extractor used in [21]. From previous research, [21], we also know that these models suffer more from domain shift effects for the data we used in the experiments.

Chapter 4

Results Evaluation

4.1 Dataset and setup

The results are generated by training and testing with the Widar3.0 dataset [31]. Widar3.0 is an open-source WiFi-CSI dataset with the presence of domain factors and their labels. This dataset is processed as close as in the methods of our research thesis used this dataset. Specifically, as the model of [21] serves as a baseline, we use the same Widar3.0 dataset as the dataset used in this thesis. 6 gestures are used, including pushpull, sweep, clap, slide, draw-O (horizontal), and draw zigzag (horizontal) [21]. For domain factors, we use all torso locations and face orientations only in the classroom environment, and user ids 10, 12, 13, 14, 15, and 16 [21]. And the size of the dataset is 4500 = 6 gestures $\times$ 5 repetitions $\times$ 5 torso locations $\times$ 5 face orientations $\times$ 6 users [21]. The Widar3.0 data is pre-processed with the method from [21] and used in this thesis.

The execution of the training was done on the GPU (Tesla V100-SXM2-16GB, core clock=1.53GHz, core count=80, memory=15.78GiB, device memory bandwidth = 836.37GiB/s) TensorFlow and other necessary libraries were used for the implementation.

4.2 Results

4.2.1 Inserting noise offline

As mentioned in Section 3.2.2, the first experiment used a different amount of false domain labels generated offline. The results of the first experiment method are shown in Figure 4.1, Figure 4.2 and Figure 4.3 for domain leave out factor user, torso location, face orientation, respectively (for detailed val-

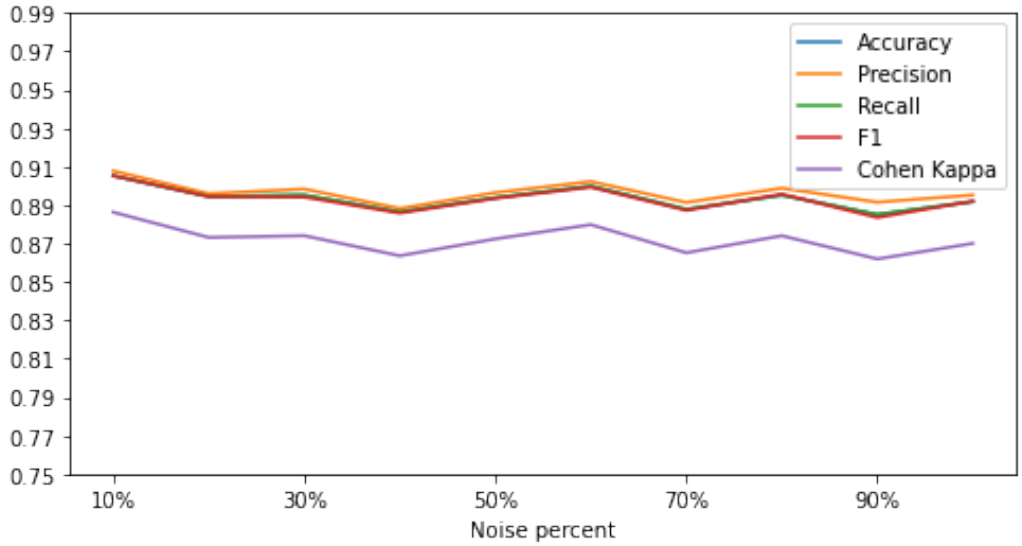

Figure 4.1: Inference metrics for first experiment on validation data with user as domain leave out factor

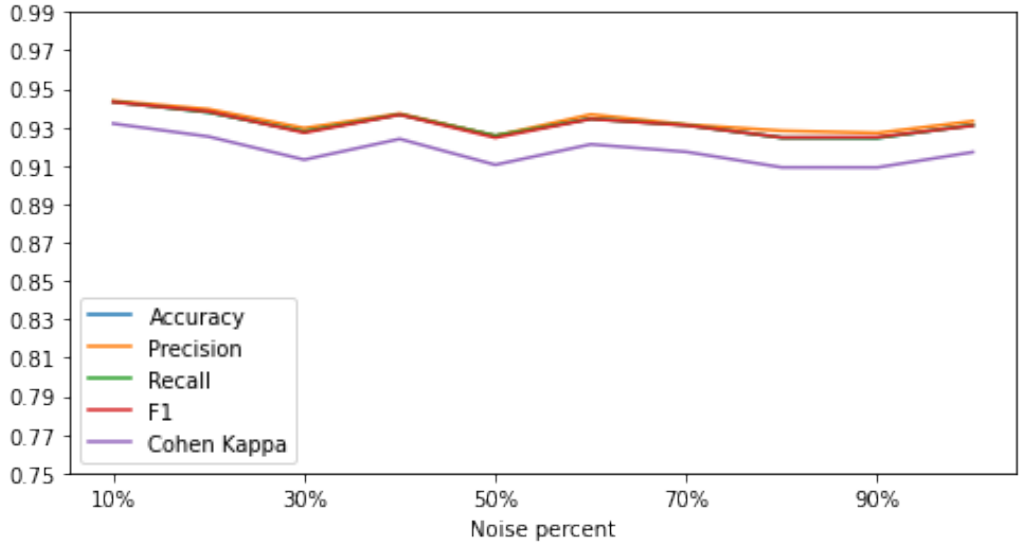

Figure 4.2: Inference metrics for first experiment on validation data with position as domain leave out factor

ues please see Table A.1, Table A.2 and Table A.3 in Appendix A). The values presented by the lines are the medians of the performance metrics across all the validation sets used for training with a certain percent of domain label noise. We could see that the validation inference metrics are not sensitive at

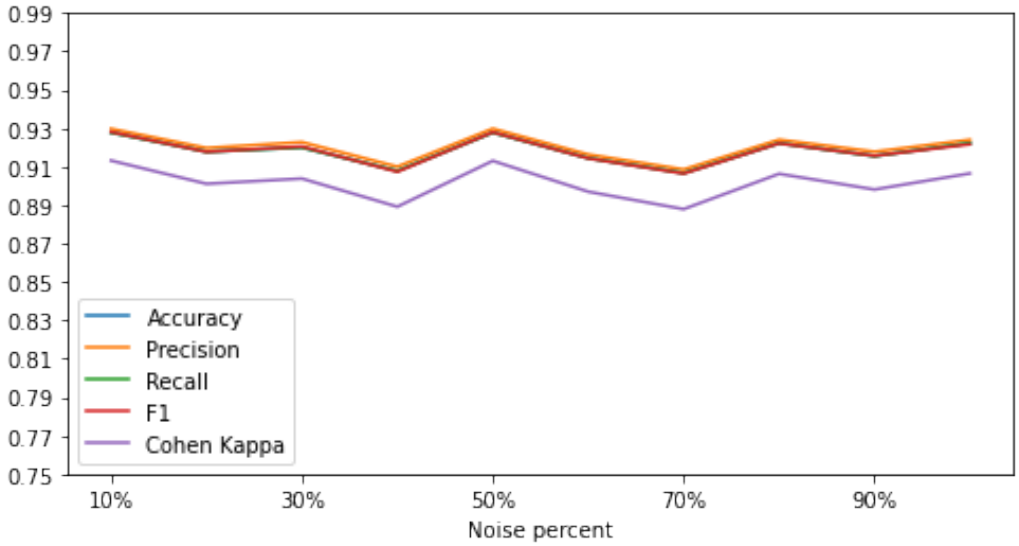

Figure 4.3: Inference metrics for first experiment on validation data with orientation as domain leave out factor

all to the changes of mislabels generated offline. The values of the metrics do not vary compared to training with clean data. For each percent of false domain labels, 5 to 6 cross-validation (depending on the domain leave-out factor) splits were performed. Hence, the at most two percent change of the values in the tables is not any significant change.

4.2.2 Inserting noise during training

As mentioned in Section 3.2.2, in the second experiment, we inserted false domain labels directly during the training into each batch. Although the amount of false domain labels per batch fed to the model is much more ensured compared with the first experiment approach, no apparent changes can be seen from the results of the first experiment (see Figure 4.1, Figure 4.2 and Figure 4.3) to the results of the second experiment shown in Figure 4.4, Figure 4.5 and Figure 4.6 (for detailed values please see tables in Appendix A). The curves are only becoming more smooth and flat, indicating that when randomness is better controlled. This proves that inserting the false domain labels offline and during the training are both valid methods.

We also reduced the latent representation and plotted them in 2-dimensional space to see whether inserting false domain labels affects the independence between the domain factors and the gestures prediction. Figure 4.7 is a visualization of an example latent representation when no false domain labels

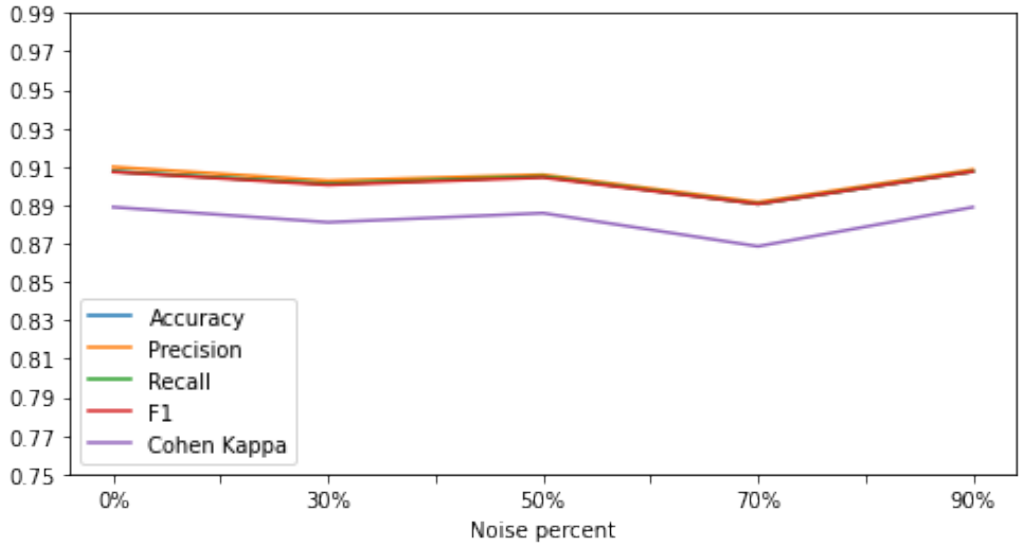

Figure 4.4: Inference metrics for second experiment on validation data with user as domain leave out factor

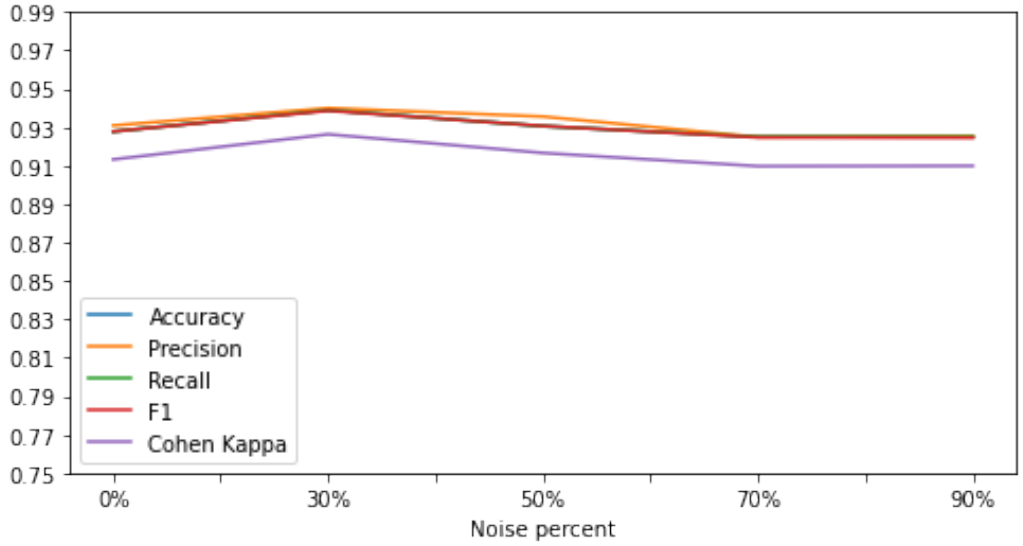

Figure 4.5: Inference metrics for second experiment on validation data with position as domain leave out factor

were used in training. We may see that although all clusters in different colors representing the gesture labels are separated well, different shapes representing the domain labels are relatively mixed but still sometimes concentrate together. This means that the model we use still could not achieve complete

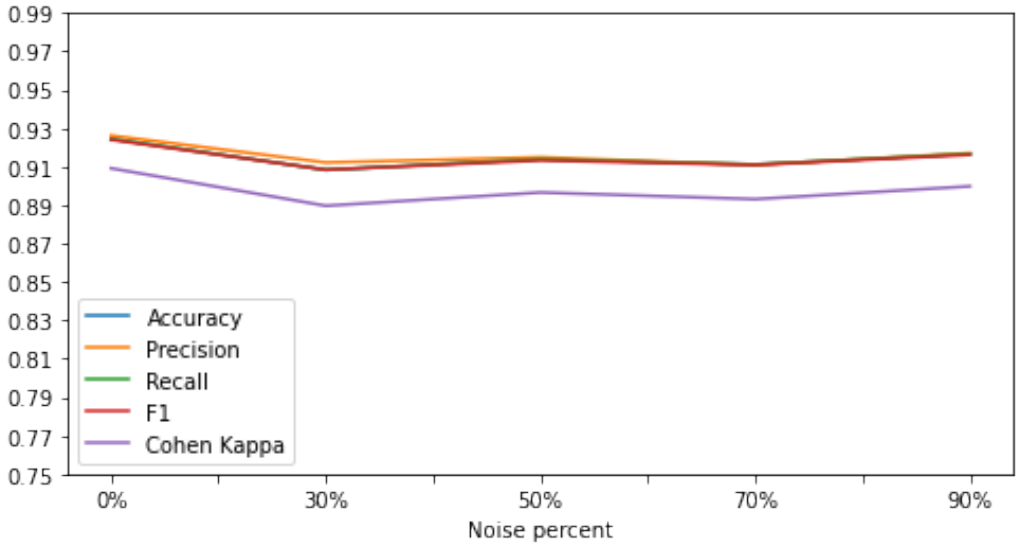

Figure 4.6: Inference metrics for second experiment on validation data with orientation as domain leave out factor

independence of the gesture prediction from domain factor effects. The data we used for Figure 4.7 is split from domain leave-out cross-validation using face orientation as the leave-out factor. For comparison, we also used the same test dataset and made all the plots for the model trained with different percentages of false domain labels. All plots are included in Appendix B. In addition, we included the plots made by the same methods on the same test data to show whether false domain labels influence the independence in the clusters of the latent representation from domain factor effects. However, no matter which false domain labels environment, the clusters remain separated to the same extent; the domain labels also remain relatively the same - mixed but sometimes with small concentrations. From the perspective of domain shift brought by the test data, this indicates that inserting a fixed amount of domain labels in the current model neither improves nor decreases the domain independence of the latent representation.

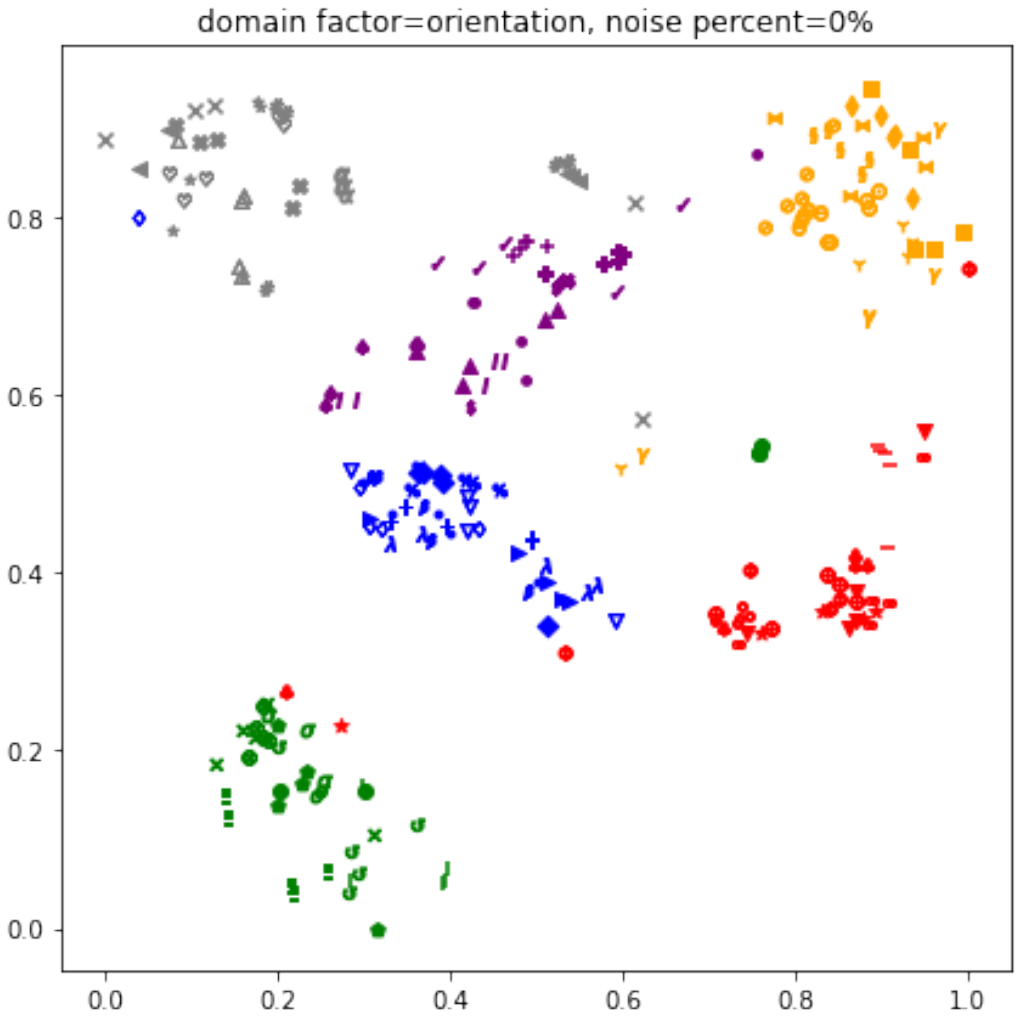

Figure 4.7: Latent representation trained without false domain labels inserted

4.2.3 Increase weight and depth of domain discriminator

Following the method mentioned in Section 3.2.2, we further increased the size of the domain discriminator (see Section 3.1.3), hypothesizing that the reason why the evaluation metrics are insensitive to the false domain labels is because the domain discriminator has little influence on the model. However, increasing the number of layers from 2 to 4 or even 6 did not bring much change to the training, as shown by the unchanged results. Results are shown in Figure 4.8 and Figure 4.9 (for detailed values see Table A.7 and Table A.8). Therefore, we only include the results using face orientation as

the domain-leave-out factor in reducing repetition.

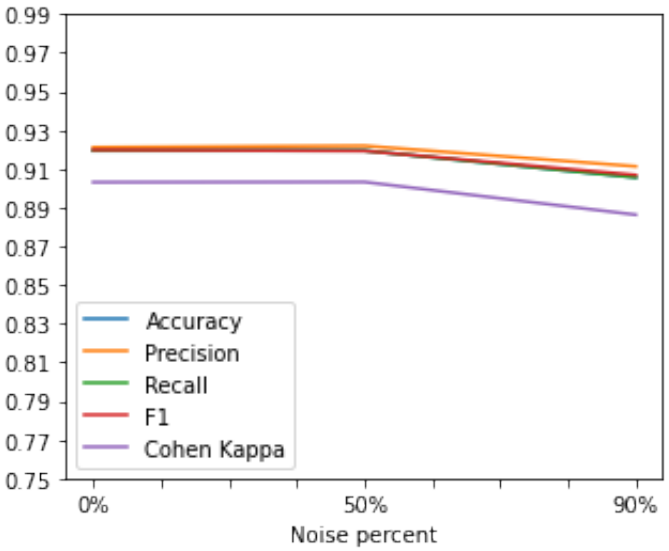

Figure 4.8: Inference metrics on validation data with face orientation as domain leave out factor using domain discriminators of 4 feedforward layers

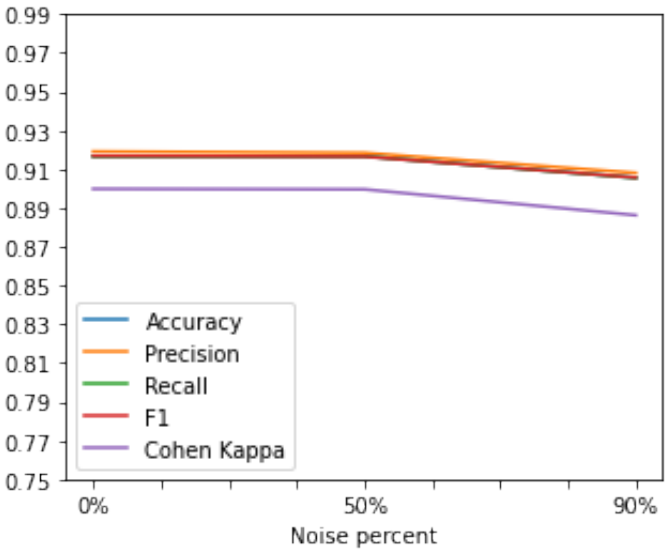

Figure 4.9: Inference metrics on validation data with face orientation as domain leave out factor using domain discriminators of 6 feedforward layers

We also experimented with increasing the weight of the domain discriminator illustrated in equation 3.1. However, the results showed that the

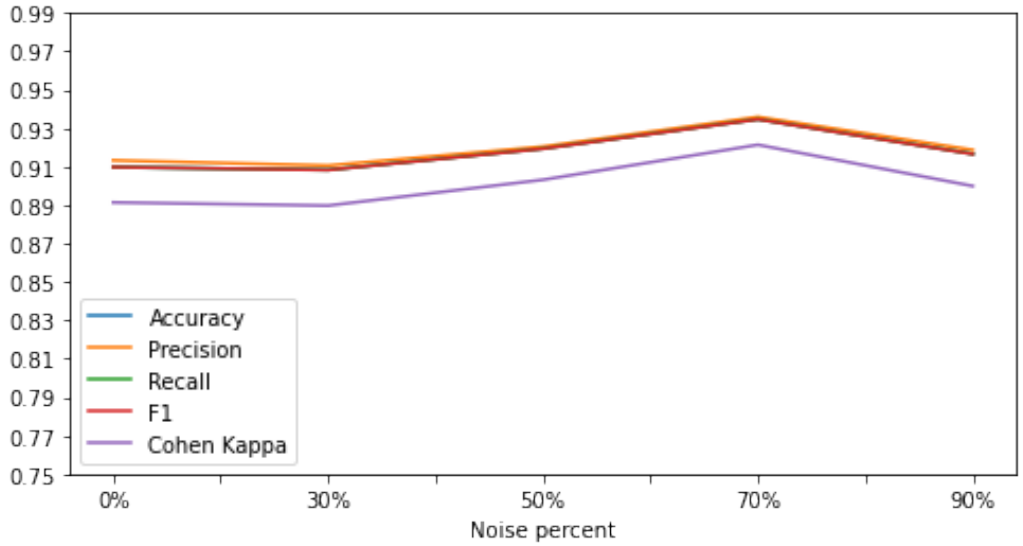

Figure 4.10: Inference metrics on validation data with face orientation as domain leave out factor using domain discriminators of 6 feedforward layers and weight of 0.00001

weight we used for all the previous experiments (0.000001) is already the highest weight with which the eventual loss can always converge. Therefore, we tried to reduce this weight by a division step of 10 and on the weight of 0.00001. Using this higher weight, three of the experiments already could not converge. However, we still calculate the validation evaluation metrics on the rest of the experiment runs. The results in Figure 4.10 shows that the increased weight does not change the influence of the false domain labels on the model's behavior (for more detailed values see Table A.9). Any weight lower than 0.00001 could not give convergent training results.

4.2.4 Change architecture of feature extractor

Following the last method we mentioned in Section 3.2.2, we changed the backbone architecture with the pre-defined Xception and ResNet50 models from Keras. For both backbone architectures, we trained the model with the highest possible domain discriminator weight (0.000001 for Xception, 0.00000001 for ResNet50). In Figure 4.11 (for exact values see Table A.10), we see that the performance using Xception as the backbone drops for all noise percents compared with training using the backbone from [21], but the values are still very consistent with the change of noise percents. Since using the Xception model as the backbone model is known to adapt poorly

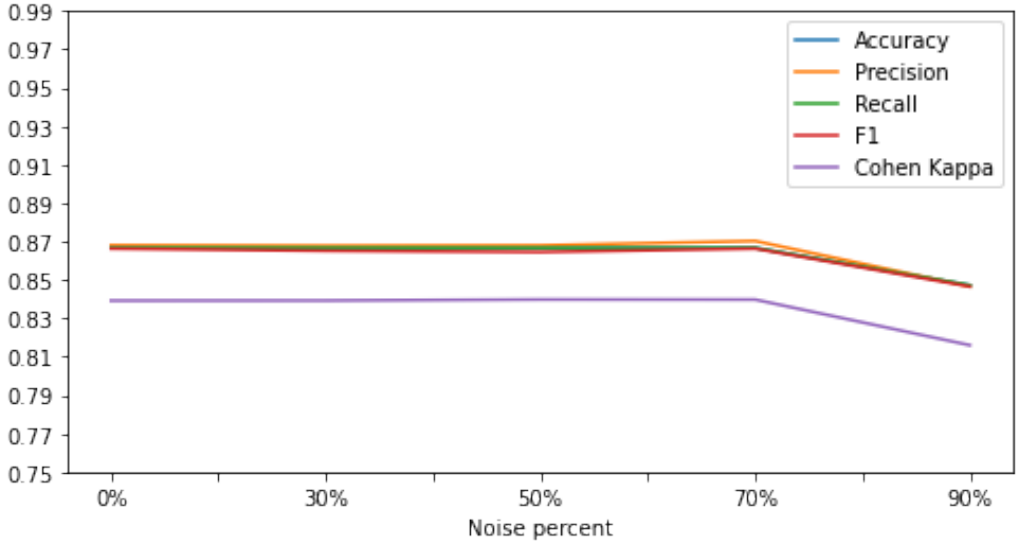

Figure 4.11: Inference metrics on validation data with face orientation as domain leave out factor, domain discriminators of 6 feedforward layers, and Xception as backbone

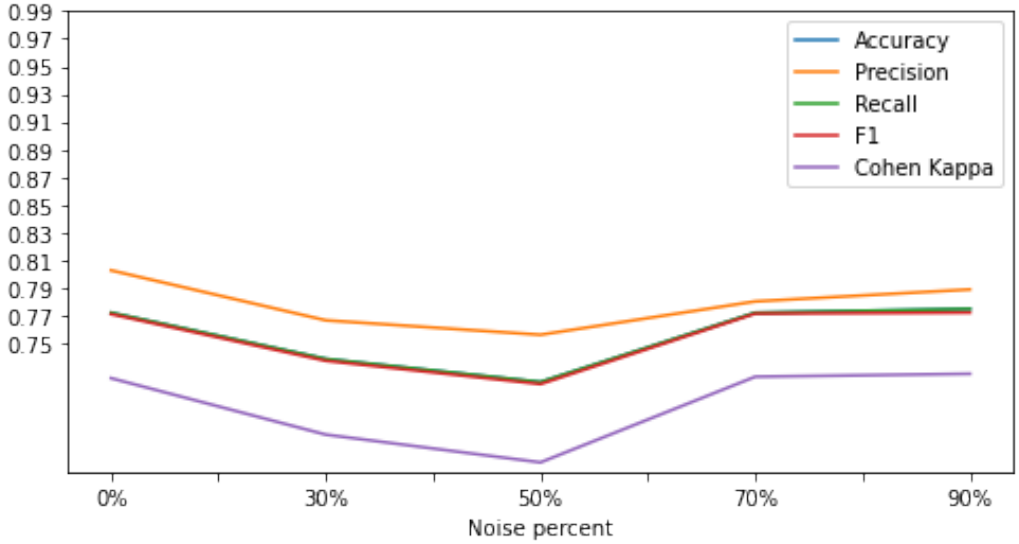

Figure 4.12: Inference metrics on validation data with face orientation as domain leave out factor, domain discriminators of 6 feedforward layers, and ResNet50 as backbone

against domain shift, we expect to see more significant changes when we vary the number of false labels that are different from those already used in training. Since using a less optimal backbone still give consistent results, and the weight and depth of the domain discriminator were affecting the model to the most extent, this may indicate that the current domain discriminator steers the model's training using some unknown heuristic without the help of correct domain labels.

In Figure 4.12 (for exact values see Table A.11), however, we could see a more noticeable decrease and a more significant deviation in all the evaluations metrics, and when noise percent is 30% to 50%, this may suggest that when using ResNet50 as the backbone, the model is affected more significantly when smaller amount false domain labels are mixed with actual labels, however, the model starts to rely less on the actual domain labels when the number of false domain labels exceeds the accurate domain labels. However, whether this insight can be transferred to different models. However, it primarily means that the architecture of the feature extractor could influence how the model reacts to false domain labels.

4.3 Discussion

All of the experiments conducted in this research showed that the training of the model we used seemingly was not affected by false information in the domain labels. Changing the weight and depth of the domain discriminator gave the same results too. Therefore, no direct insights can be derived to conduct the definition of reward function for the reinforcement learning agent to achieve domain adaptation. We will discuss the potential reasons for such insensitivity of the model.

One may argue that the model's insensitivity to the false domain labels is because the noise injection potentially only caused permutation (changes of positions) but not meaningful changes of values for the weight magnitudes in the domain discriminator. The domain discriminator is a multilayer perceptron (MLP) component that does not assume any spatial relationships between the features; therefore, the domain discriminator is permutation invariant, meaning that any permutations to shuffle the input data would not affect the model's training [27]. It also follows that the relocation of the weight magnitudes to different neurons caused by the permutation in the input data will not affect the model's output and the performance evaluation. Although we did not change the input data, it can be possible that the domain labels were shifted in an undesirable way, and this caused the similar permutation of the weight magnitudes and the following unchanged

inference performances.

Hypothetically, one situation that may position the same weight magnitudes differently is when the same domain labels are only replaced by the same false domain labels. In other words, the distribution domain labels may not be changed essentially, but the labels are only annotated differently, especially in the training processes that only include part of the domain factors. However, in the experiment setting, we made sure to involve as much randomness as possible in the noise injection procedure to prevent this dependency between the true labels and the certain false domain labels to assign. In every batch, the instances changed to wrong domain labels are selected uniformly with a fixed probability. For each instance, the false domain label is selected with the same probability from all the domain labels, excluding the true label. In this way, no randomness is considered the true domain label. To realize this, we used a uniform distribution function implemented by TensorFlow [12] with no seeds in order to diversify the random assignment. Theoretically, with enough iterations per epoch of such random noise injection, the likelihood that permutation invariance of the MLP being the cause of the unchanged performance is very low. However, apart from setting no seeds, it is still worthwhile to keep a record of the label assignment in future experiments and ensure that instances with the same true domain label would indeed be assigned various different false domain labels.

Another argument for the potential reason for the unchanged inference performance is that the false domain labels may interfere with the adversarial training and demotivate the model to maximize the domain loss. Compared with training using clean domain labels, noisy domain labels are usually assumed to lead to higher loss since the model learns the pattern first, and the false labels are wrongly associated with the pattern of the data [19]. Moreover, since the false labels are randomly assigned while clean domain label indices belong to a certain range, they are more likely to be underrepresented classes which will also lead to higher losses [25]. Therefore, the false domain labels may already cause a higher loss of the domain classification at the very initial stage of training. At the early stage of the training, a standard deep learning model is known to first learn from "easier" instances which lead to smaller loss values [2]. However, the goal is set reversely for adversarial training on the domain discriminator; this means the model will first adapt to the instances that lead to higher loss. Note that the loss of the domain discriminator is set to be maximized; therefore, if a larger loss value of the domain discriminator is already reached at the early stage of training, the model would not need to drastically increase the loss of the domain discriminator to converge. Then the model will not learn sufficiently from the domain information, which reduces the influence of the domain label noise

on the model and causes the unchanged evaluation performance regardless of the noise levels. In this thesis, no careful check was done on the differences of the initial loss values with respect to different domain label noise levels. Given that the above-mentioned hypothesis may be the reason we see our results, more tests, for example, plotting the changes of the domain classification loss, should be done in future research. If such a case is the reason, methods such as reducing the learning rate and (or) the batch size in early training can be tested further.

According to the results of experiments, limitations of the methods in this thesis may also cause insensitivity to false domain labels of the model. The results of the experiments can provide a more straightforward discussion basis for the following potential reasons.

The data we used can already contain false domain labels. As the authors of [21] mentioned, the lack of domain information on currently existing data, for example, the Widar3.0 data we used, made correct domain leave out cross-validation on the exact factors hard to realize. Therefore, in both training/validation and test data, we may already have a certain amount of factors from other domains present, which mixed with the mislabels we introduced and reduced the impact of the mislabels.

Moreover, this insensitivity may be related to the architecture of the model. The architecture of the current domain discriminator is not optimized. Compared with the complexity and depth of the feature extractor, the domain discriminator is still a very small part of the model. In this research, the deepest version of it is only a Multilayer Perceptron (MLP) with six dense layers. Such components may not have enough complexity to pick up the necessary domain-factor-related feature information needed for backpropagation. Therefore, the lack of complexity may cause the model to ignore the false domain label information. The feature extractor based on the efficientNet network is a newer, more advanced neural network. In [21], this network has already shown good inference performance on domain shift. Due to this, the changes in domain labels, which only influence a very small part of the model, may not be enough to make it suffer from failures of domain adaptation.

The experiment to insert false domain labels is only a very simplistic design. Although it is reasonable to assume that a simple but straightforward experiment should already affect the model, advancements in current design may give more insightful results. For example, varying the amount of domains in a different manner (linear and non-linear, drastic and gradual) during the training before the model generalizes well may add more influence on the model, therefore, may provide more variant results with respect to the changes of noise.

Chapter 5

Conclusion and future work

In this thesis, we experimented on an adversarial model with a traditional domain discriminator used by the authors of [21] for extracting the domain-independent representation. The goal is to see whether and how this model reacts to the false domain labels fed to the domain discriminator, which to some extent similar to an auto-labeling procedure for unknown domain factors, in order to find preliminary insights on how to maintain the independence of the latent representation from the domain factors effect while the auto-labeling happens.

However, all the experiments we performed failed to show very insightful results as the model did not react to the domain label noises we inserted in several different ways, including inserting the noise both offline and during the training, altering the weight and depth of the domain discriminator and changing the architecture of the backbone. We further discussed the possible reason for this, including the permutation invariance of the domain discriminator, lack of subsequent maximizing domain loss, issues with the data we used, the lack of complexity in the domain discriminator, the architecture of the feature extractor, and the inflexible inserting of false labels. Hence, this research may provide a basis for future research continuation.

Necessary future work has to be done addressing the limitation of this research. For example, one could test if false domain labels interfere with adversarial training and try reducing the focus of the model on the false domain labels leading to high losses; one could try changing the architecture of the domain discriminator more extensively in order to increase its influence on the entire model. Using a backbone structure that suffers more from domain shift issue could possibly provide some quick insight too; however, if newer networks are needed for better performance, how transferable this evidence is to the newer neural networks need to be investigated. Once more insights are gathered from future research with respect to the changes

in domain labels noise, a more adaptive method to change the number of domain labels during the training (e.g., for each epoch) can also be applied in order better to mimic the possible influence of the reinforcement learning agent.

Bibliography

[1] Hasmath Farhana Thariq Ahmed, Hafisoh Ahmad, and CV Aravind. Device free human gesture recognition using wi-fi csi: A survey. *Engineering Applications of Artificial Intelligence*, 87:103281, 2020.

[2] Devansh Arpit, Stanisław Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International conference on machine learning*, pages 233–242. PMLR, 2017.

[3] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[4] T Eady. Tesla's deep learning at scale: Using billions of miles to train neural networks. *Towards Data Science*, 7, 2019.

[5] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[7] Yu Gu, Xiang Zhang, Yantong Wang, Meng Wang, Huan Yan, Yusheng Ji, Zhi Liu, Jianhua Li, and Mianxiong Dong. Wigrunt: Wifi-enabled gesture recognition using dual-attention network. *IEEE Transactions on Human-Machine Systems*, 2022.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[10] Wenjun Jiang, Chenglin Miao, Fenglong Ma, Shuochao Yao, Yaqing Wang, Ye Yuan, Hongfei Xue, Chen Song, Xin Ma, Dimitrios Koutsonikolas, et al. Towards environment independent device free human activity recognition. In *Proceedings of the 24th annual international conference on mobile computing and networking*, pages 289–304, 2018.

[11] Hua Kang, Qian Zhang, and Qianyi Huang. Context-aware wireless-based cross-domain gesture recognition. *IEEE Internet of Things Journal*, 8(17):13503–13515, 2021.

[12] Nikhil Ketkar. Introduction to tensorflow. In *Deep learning with Python*, pages 159–194. Springer, 2017.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Jianwei Liu, Jinsong Han, Feng Lin, and Kui Ren. Adversary helps: Gradient-based device-free domain-independent gesture recognition. *arXiv preprint arXiv:2004.03961*, 2020.

[15] Yongsen Ma, Sheheryar Arshad, Swetha Muniraju, Eric Torkildson, Enrico Rantala, Klaus Doppler, and Gang Zhou. Location-and person-independent activity recognition with wifi, deep neural networks, and reinforcement learning. *ACM Transactions on Internet of Things*, 2(1):1–25, 2021.

[16] Yongsen Ma, Gang Zhou, and Shuangquan Wang. Wifi sensing with channel state information: A survey. *ACM Computing Surveys (CSUR)*, 52(3):1–36, 2019.

[17] Yongsen Ma, Gang Zhou, Shuangquan Wang, Hongyang Zhao, and Woosub Jung. Signfi: Sign language recognition using wifi. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(1):1–21, 2018.

[18] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[19] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[20] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[21] 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)*, pages 527–532. IEEE, 2022.

[22] Laurens Van Der Maaten. Barnes-hut-sne. *arXiv preprint arXiv:1301.3342*, 2013.

[23] Aditya Virmani and Muhammad Shahzad. Position and orientation agnostic gesture recognition using wifi. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 252–264, 2017.

[24] Zhengyang Wang, Sheng Chen, Wei Yang, and Yang Xu. Environment-independent wi-fi human activity recognition with adversarial network. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3330–3334. IEEE, 2021.

[25] Xiaobo Xia, Tongliang Liu, Bo Han, Mingming Gong, Jun Yu, Gang Niu, and Masashi Sugiyama. Sample selection with uncertainty of losses for learning with noisy labels. *arXiv preprint arXiv:2106.00445*, 2021.

[26] Yuqing Yin, Zixin Zhang, Xu Yang, Faren Yan, and Qiang Niu. Towards fully domain-independent gesture recognition using cots wifi device. *Electronics Letters*, 57(5):232–234, 2021.

[27] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *2018 International Conference on 3D Vision (3DV)*, pages 728–737. IEEE, 2018.

[28] Bin-Bin Zhang, Dongheng Zhang, Yadong Li, Yang Hu, and Yan Chen. Unsupervised domain adaptation for device-free gesture recognition. *arXiv preprint arXiv:2111.10602*, 2021.

[29] Jie Zhang, Zhanyong Tang, Meng Li, Dingyi Fang, Petteri Nurmi, and Zheng Wang. Crosssense: Towards cross-site and large-scale wifi sensing.

In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 305–320, 2018.

[30] Lei Zhang, Yixiang Zhang, and Xiaolong Zheng. Wisign: Ubiquitous american sign language recognition using commercial wi-fi devices. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–24, 2020.

[31] Yi Zhang, Yue Zheng, Kun Qian, Guidong Zhang, Yunhao Liu, Chenshu Wu, and Zheng Yang. Widar3. 0: Zero-effort cross-domain gesture recognition with wi-fi. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[32] Mingmin Zhao, Shichao Yue, Dina Katabi, Tommi S Jaakkola, and Matt T Bianchi. Learning sleep stages from radio signals: A conditional adversarial architecture. In *International Conference on Machine Learning*, pages 4100–4109. PMLR, 2017.

[33] Yue Zheng, Yi Zhang, Kun Qian, Guidong Zhang, Yunhao Liu, Chenshu Wu, and Zheng Yang. Zero-effort cross-domain gesture recognition with wi-fi. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, pages 313–325, 2019.

[34] Han Zou, Jianfei Yang, Yuxun Zhou, Lihua Xie, and Costas J Spanos. Robust wifi-enabled device-free gesture recognition via unsupervised adversarial domain adaptation. In *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–8. IEEE, 2018.

List of Figures

List of Tables

Appendix A

Appendix results in tables

Noise percent	Accuracy	Precision	Recall	F1	Cohen Kappa
0%	0.91±0.04	0.91±0.03	0.91±0.04	0.91±0.04	0.89±0.04
10%	0.91±0.06	0.91±0.05	0.91±0.06	0.91±0.06	0.89±0.07
20%	0.89±0.03	0.90±0.02	0.89±0.03	0.89±0.03	0.87±0.03
30%	0.90±0.09	0.90±0.07	0.90±0.09	0.89±0.09	0.87±0.11
40%	0.89±0.02	0.89±0.03	0.89±0.02	0.89±0.02	0.86±0.03
50%	0.89±0.04	0.90±0.04	0.89±0.04	0.89±0.04	0.87±0.05
60%	0.90±0.06	0.90±0.05	0.90±0.06	0.90±0.06	0.88±0.07
70%	0.89±0.07	0.89±0.06	0.89±0.07	0.89±0.07	0.87±0.08
80%	0.90±0.04	0.90±0.03	0.90±0.04	0.90±0.04	0.87±0.04
90%	0.89±0.05	0.89±0.05	0.89±0.05	0.88±0.05	0.86±0.06
99.9%	0.89±0.05	0.90±0.05	0.89±0.05	0.89±0.05	0.87±0.06

Table A.1: Inference metrics for first experiment on validation data with user as domain leave out factor

Noise percent	Accuracy	Precision	Recall	F1	Cohen Kappa
0%	0.93±0.02	0.93±0.02	0.93±0.02	0.93±0.02	0.91±0.03
10%	0.94±0.04	0.94±0.04	0.94±0.04	0.94±0.04	0.93±0.05
20%	0.94±0.02	0.94±0.02	0.94±0.02	0.94±0.02	0.93±0.02
30%	0.93±0.06	0.93±0.06	0.93±0.06	0.93±0.06	0.91±0.07
40%	0.94±0.03	0.94±0.03	0.94±0.03	0.94±0.03	0.92±0.04
50%	0.93±0.02	0.93±0.02	0.93±0.02	0.92±0.02	0.91±0.02
60%	0.93±0.03	0.94±0.04	0.93±0.03	0.93±0.03	0.92±0.04
70%	0.93±0.03	0.93±0.03	0.93±0.03	0.93±0.03	0.92±0.04
80%	0.92±0.02	0.93±0.02	0.92±0.02	0.92±0.02	0.91±0.02
90%	0.92±0.06	0.93±0.06	0.92±0.06	0.92±0.06	0.91±0.07
99.9%	0.93±0.02	0.93±0.02	0.93±0.02	0.93±0.02	0.92±0.03

Table A.2: Inference metrics for first experiment on validation data with torso location as domain leave out factor

Noise percent	Accuracy	Precision	Recall	F1	Cohen Kappa
0%	0.92±0.02	0.93±0.02	0.92±0.02	0.92±0.02	0.91±0.03
10%	0.93±0.04	0.93±0.04	0.93±0.04	0.93±0.04	0.91±0.05
20%	0.92±0.04	0.92±0.04	0.92±0.04	0.92±0.04	0.90±0.04
30%	0.92±0.03	0.92±0.04	0.92±0.03	0.92±0.03	0.90±0.04
40%	0.91±0.03	0.91±0.03	0.91±0.03	0.91±0.03	0.89±0.03
50%	0.93±0.02	0.93±0.02	0.93±0.02	0.93±0.02	0.91±0.02
60%	0.91±0.02	0.92±0.02	0.91±0.02	0.91±0.02	0.90±0.03
70%	0.91±0.02	0.91±0.02	0.91±0.02	0.91±0.02	0.89±0.02
80%	0.92±0.04	0.92±0.04	0.92±0.04	0.92±0.04	0.91±0.04
90%	0.92±0.03	0.92±0.02	0.92±0.03	0.92±0.03	0.90±0.03
99.9%	0.92±0.02	0.92±0.02	0.92±0.02	0.92±0.02	0.91±0.03

Table A.3: Inference metrics for first experiment on validation data with face orientation as domain leave out factor

Noise percent	Accuracy	Precision	Recall	F1	Cohen Kappa
0%	0.91±0.04	0.91±0.03	0.91±0.04	0.91±0.04	0.89±0.04
30%	0.90±0.07	0.90±0.07	0.90±0.07	0.90±0.07	0.88±0.08
50%	0.90±0.02	0.91±0.02	0.90±0.02	0.90±0.02	0.89±0.03
70%	0.89±0.03	0.89±0.02	0.89±0.03	0.89±0.03	0.87±0.03
90%	0.91±0.04	0.91±0.04	0.91±0.04	0.91±0.04	0.89±0.05

Table A.4: Inference metrics for second experiment on validation data with user as domain leave out factor

Noise percent	Accuracy	Precision	Recall	F1	Cohen Kappa
0%	0.93±0.02	0.93±0.02	0.93±0.02	0.93±0.02	0.91±0.03
30%	0.94±0.03	0.94±0.03	0.94±0.02	0.94±0.03	0.93±0.03
50%	0.93±0.04	0.94±0.03	0.93±0.04	0.93±0.04	0.92±0.04
70%	0.93±0.03	0.92±0.03	0.92±0.03	0.92±0.03	0.91±0.04
90%	0.93±0.02	0.93±0.03	0.93±0.03	0.92±0.03	0.91±0.03

Table A.5: Inference metrics for second experiment on validation data with torso location as domain leave out factor

Noise percent	Accuracy	Precision	Recall	F1	Cohen Kappa
0%	0.92±0.02	0.93±0.02	0.92±0.02	0.92±0.02	0.91±0.03
30%	0.91±0.03	0.91±0.03	0.91±0.03	0.91±0.03	0.89±0.04
50%	0.91±0.02	0.91±0.01	0.91±0.02	0.91±0.02	0.90±0.02
70%	0.91±0.05	0.91±0.05	0.91±0.05	0.91±0.05	0.89±0.06
90%	0.92±0.04	0.92±0.03	0.92±0.04	0.92±0.04	0.90±0.04

Table A.6: Inference metrics for second experiment on validation data with face orientation as domain leave out factor

Noise percent	Accuracy	Precision	Recall	F1	Cohen Kappa
0%	0.92±0.02	0.92±0.02	0.92±0.02	0.92±0.02	0.90±0.02
50%	0.92±0.03	0.92±0.04	0.92±0.03	0.92±0.03	0.90±0.04
90%	0.91±0.02	0.91±0.02	0.91±0.02	0.91±0.02	0.89±0.02

Table A.7: Inference metrics on validation data with face orientation as domain leave out factor using domain discriminators of 4 feedforward layers

Noise percent	Accuracy	Precision	Recall	F1	Cohen Kappa
0%	0.92±0.10	0.92±0.09	0.92±0.10	0.92±0.10	0.90±0.12
50%	0.92±0.04	0.92±0.04	0.92±0.04	0.92±0.04	0.90±0.05
90%	0.91±0.04	0.91±0.04	0.91±0.04	0.91±0.04	0.89±0.05

Table A.8: Inference metrics on validation data with face orientation as domain leave out factor using domain discriminators of 6 feedforward layers

Noise percent	Accuracy	Precision	Recall	F1	Cohen Kappa
0%	0.91±0.01	0.91±0.01	0.91±0.01	0.91±0.01	0.89±0.02
30%	0.91±0.03	0.91±0.03	0.91±0.03	0.91±0.03	0.89±0.03
50%	0.92±0.03	0.92±0.03	0.92±0.03	0.92±0.03	0.90±0.04
70%	0.93±0.03	0.94±0.03	0.93±0.03	0.93±0.03	0.92±0.04
90%	0.92±0.01	0.92±0.01	0.92±0.01	0.92±0.01	0.90±0.01

Table A.9: Inference metrics on validation data with face orientation as domain leave out factor using domain discriminators of 6 feedforward layers and weight of 0.00001

Noise percent	Accuracy	Precision	Recall	F1	Cohen Kappa
0%	0.87±0.07	0.87±0.07	0.87±0.07	0.87±0.07	0.84±0.08
30%	0.87±0.03	0.87±0.03	0.87±0.03	0.87±0.03	0.84±0.04
50%	0.87±0.02	0.87±0.03	0.87±0.02	0.86±0.03	0.84±0.03
70%	0.87±0.07	0.87±0.07	0.87±0.07	0.87±0.07	0.84±0.08
90%	0.87±0.04	0.87±0.04	0.87±0.04	0.87±0.04	0.84±0.05

Table A.10: Inference metrics on validation data with face orientation as domain leave out factor, domain discriminators of 6 feedforward layers, and Xception as backbone

Noise percent	Accuracy	Precision	Recall	F1	Cohen Kappa
0%	0.77±0.14	0.80±0.13	0.77±0.14	0.77±0.15	0.73±0.16
30%	0.73±0.13	0.77±0.11	0.73±0.13	0.74±0.13	0.67±0.16
50%	0.72±0.23	0.76±0.17	0.72±0.23	0.72±0.22	0.66±0.27
70%	0.77±0.04	0.78±0.04	0.77±0.04	0.77±0.04	0.73±0.05
90%	0.77±0.09	0.79±0.10	0.77±0.09	0.77±0.10	0.73±0.11

Table A.11: Inference metrics on validation data with face orientation as domain leave out factor, domain discriminators of 6 feedforward layers, and Resnet50 as backbone

Appendix B

Appendix of t-SNE plots

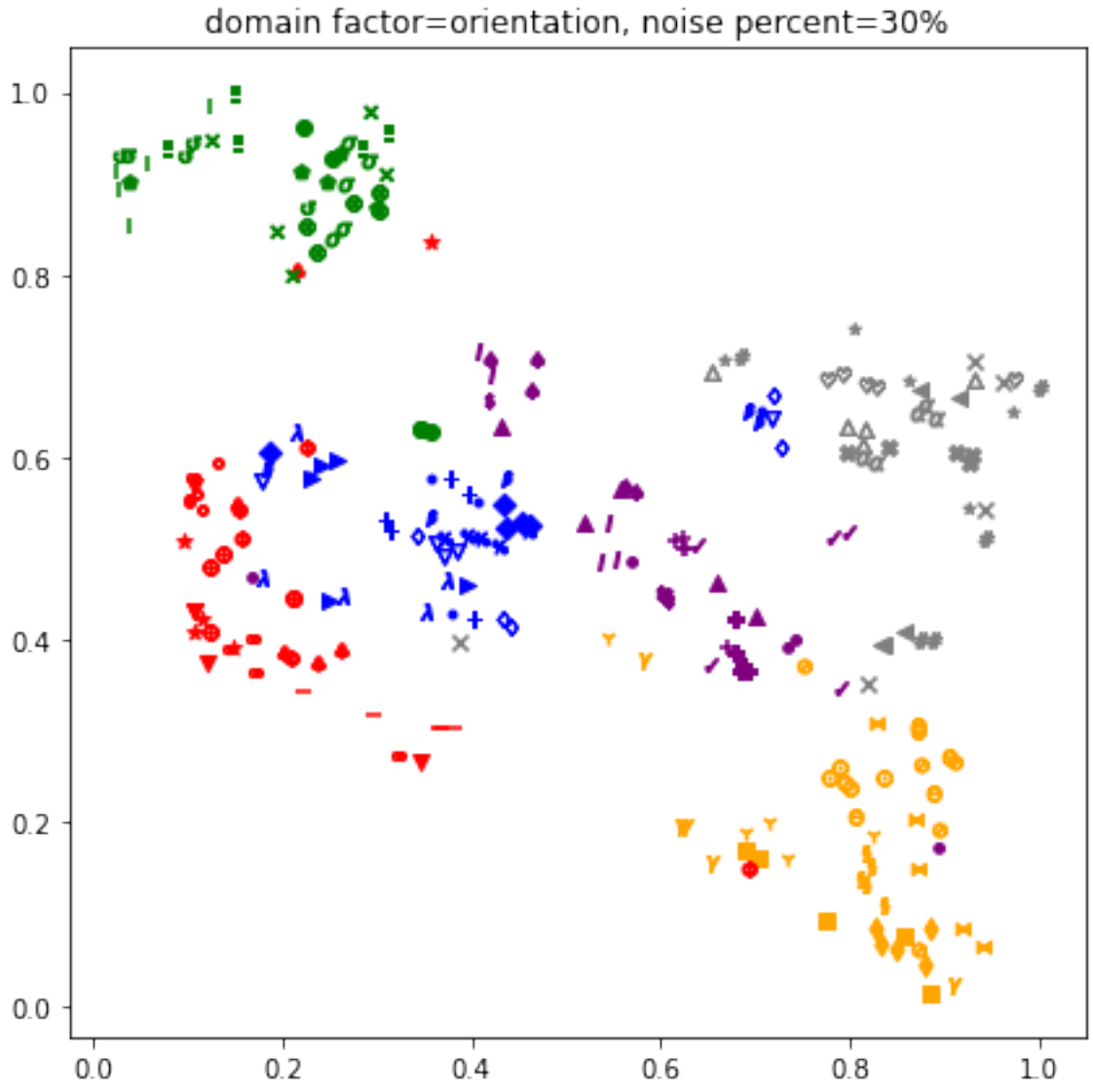

Figure B.1: Latent representation trained with 30% domain labels inserted

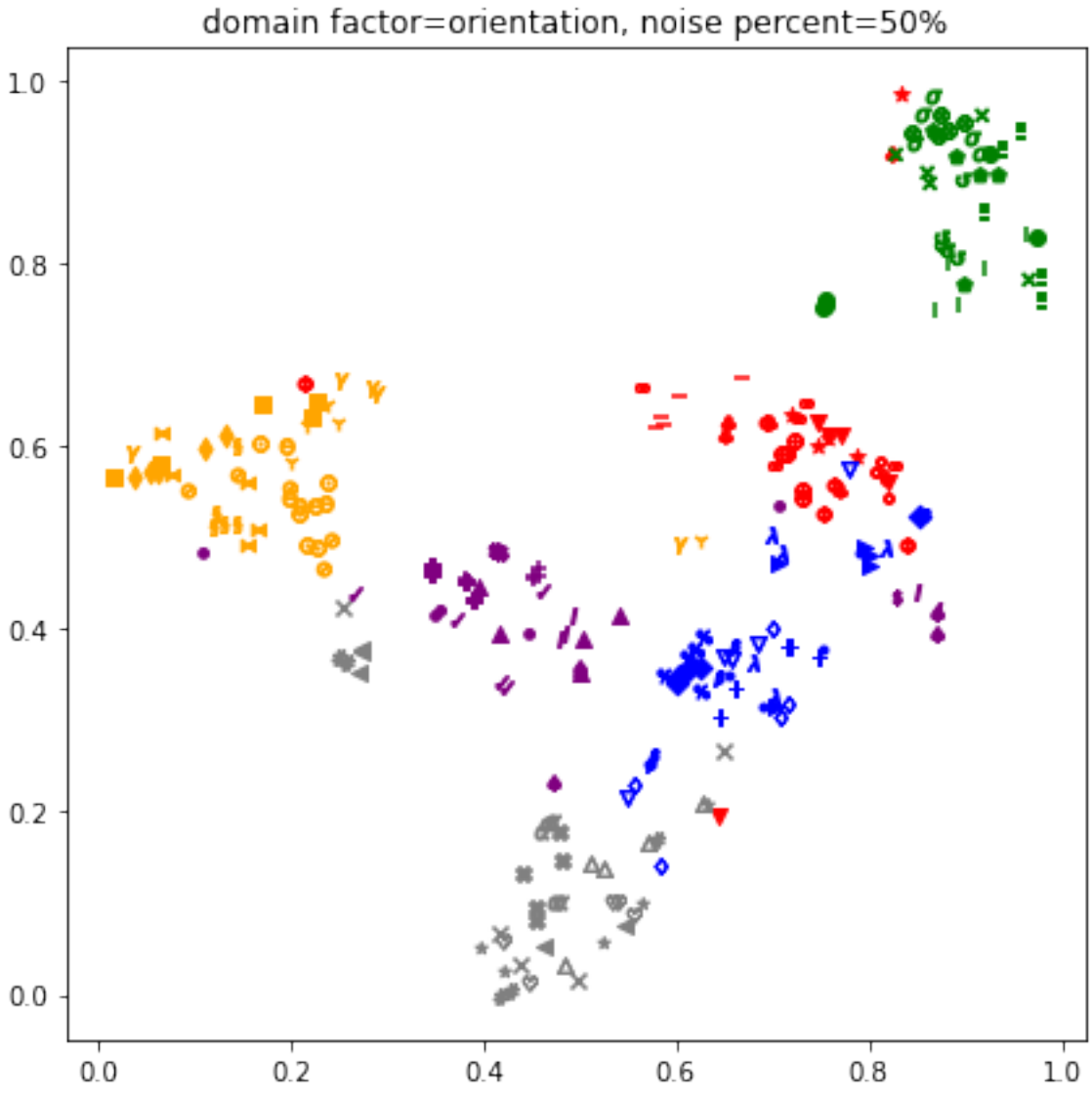

Figure B.2: Latent representation trained with 30% domain labels inserted

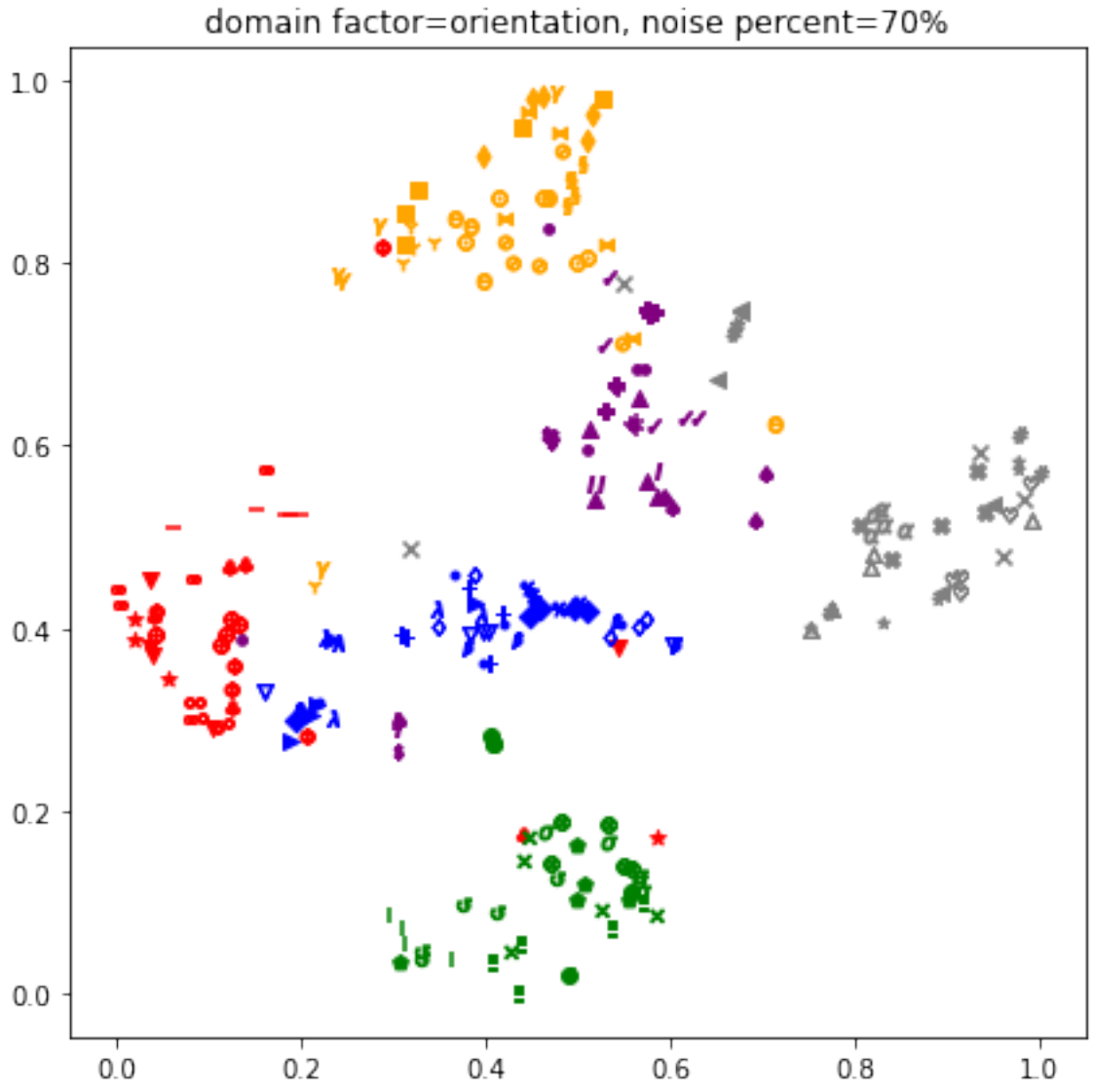

Figure B.3: Latent representation trained with 30% domain labels inserted

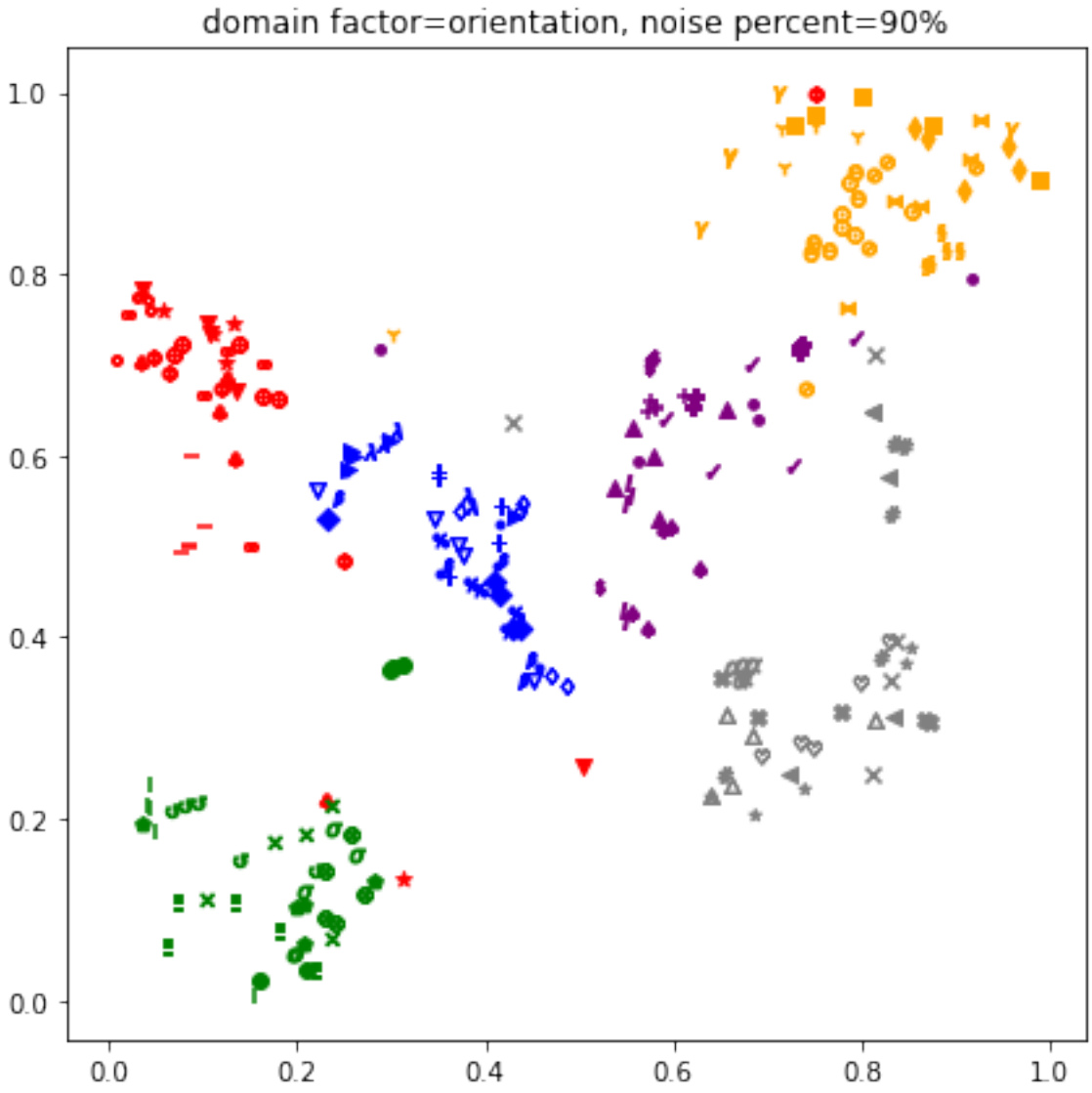

Figure B.4: Latent representation trained with 30% domain labels inserted