TRANSFER LEARNING PERFORMANCE USING DATASET SIMILARITY ON REALTIME CLASSIFICATION

PREDICTING TRANSFER LEARNING PERFORMANCE USING DATASET SIMILARITY FOR TIME SERIES CLASSIFICATION OF HUMAN ACTIVITY RECOGNITION

BY

RYAN CLARK, B.Eng.

A THESIS

SUBMITTED TO THE DEPARTMENT OF SCHOOL OF BIOMEDICAL ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2020)	McMaster University
(School of Biomedical Engineering)	Hamilton, Ontario, Canada

TITLE:	Predicting Transfer Learning Performance Using Dataset Similarity for Time Series Classification of Human Activity Recognition
AUTHOR:	Ryan Clark B.Eng. (Engineering Physics), McMaster University, Hamilton, Canada
SUPERVISOR:	Dr. Thomas E. Doyle
NUMBER OF PAGES:	xv, 98

Lay Abstract/ Thesis statement

Over the past decade, advances in computational power and increases in data quantity have made deep learning a useful method of complex pattern recognition and classification in data. There is a growing desire to be able to use these complex algorithms on smaller quantities of data. To achieve this, a deep learning model is first trained on a larger dataset and then retrained on the smaller dataset; this is called transfer learning. For transfer learning to be effective, there needs to be a level of similarity between the two datasets so that properties from larger dataset can be learned and then refined using the smaller dataset. Therefore, it is of great interest to understand what level of similarity exists between the two datasets. The goal of this research is to provide a similarity metric between two time series classification datasets so that potential performance gains from transfer learning can be better understood. The measure of similarity between two time series datasets presents a unique challenge due to the nature of this data. To address this challenge an encoder approach was implemented to transform the time series data into a form where each signal example can be compared against one another. In this thesis, different similarity metrics were evaluated and correlated to the performance of a deep learning model allowing the prediction of how effective transfer learning may be when applied.

Abstract

Deep learning is increasingly becoming a viable way of classifying all types of data. Modern deep learning algorithms, such as one dimensional convolutional neural networks, have demonstrated excellent performance in classifying time series data because of the ability to identify time invariant features. A primary challenge of deep learning for time series classification is the large amount of data required for training and many application domains, such as in medicine, have challenges obtaining sufficient data. Transfer learning is a deep learning method used to apply feature knowledge from one deep learning model to another; this is a powerful tool when both training datasets are similar and offers smaller datasets the power of more robust larger datasets. This makes it vital that the best source dataset is selected when performing transfer learning and presently there is no metric for this purpose.

In this thesis a metric of predicting the performance of transfer learning is proposed. To develop this metric this research will focus on classification and transfer learning for human-activity-recognition time series data. For general time series data, finding temporal relations between signals is computationally intensive using non-deep learning techniques. Rather than time-series signal processing, a neural network autoencoder was used to first transform the source and target datasets into a time independent feature space. To compare and quantify the suitability of transfer learning datasets, two metrics were examined: i) average embedded signal from each dataset was used to calculate the distance between each datasets centroid, and ii) a Generative Adversarial Network (GAN) model was trained and the discriminator portion of the GAN is then used to assess the dissimilarity between source and target. This thesis measures a correlation between the distance between two dataset and their similarity, as well as the ability for a GAN to discriminate between two datasets and their similarity. The discriminator metric, however, does suffer from an upper limit of dissimilarity. These metrics were then used to predict the success of transfer learning from one dataset to another for the purpose of general time series classification.

To my family

Acknowledgements

I would like to thank my supervisor, Dr. Thomas Doyle, for his guidance and support throughout my masters. Dr. Doyle has contributed greatly to my project through his expertise and knowledge in the field. I would also like to acknowledge Mohammadreza (Reza) Heydarian for sharing his technical expertise with me throughout my academic career.

Next, I would like to thank my committee members Dr. Michael Noseworthy and Dr. Carol DeMatteo for their contributions. The two of them introduced me to the projected and trusted me with finding the solutions present in this thesis.

I would also like to recognize my family (Mom, Dad, Sara) who supported me in pursuing my academic interests, as well as friends (Nik, Mason, Marco, Luis and Aaron) who always ensured that I maintained balance in my life. I would also like to recognize Sarah Cooper, who has always given me unrelenting support when I needed it most.

Lastly, I would like to acknowledge McMaster University, my home for the past seven years, for funding my project and research.

Contents

List of Figures

List of Tables

Notation, Definitions, and Abbreviations

Abbreviations

ABC	Area Between Curves
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
LSTM	Long Short Term Memory
TL	Transfer Learning

Chapter 1

Introduction

As the computational power of everyday devices increases, there is a growing desire for use of deep learning algorithms to solve more and more classification problems. Deep learning is attractive because it requires minimal data pre-processing; however, it does require large datasets for its training. This requirement is particularly problematic when data is limited, such as in areas like precision medicine, or space crew medical support. To address this challenge, transfer learning provides an avenue for increasing deep learning performance when a lack of data is present.

In this thesis, a new method of predicting the suitability, or most suitable, of the selected dataset(s) as the source dataset for the transfer learning target is proposed. Two new methods of measuring similarities are presented along with a method of transforming datasets of time-series data into datasets of time-invariant embeddings. One method then measures the Manhattan distance between centroids of the embedding clusters. The second method uses a discriminator to learn which dataset an example comes from, and the accuracy of the discriminator measures the similarity. In the following sections, the background and motivation for developing such a metric is introduced. The main contributions are summarized, and an outline of the thesis is presented.

1.1 Problem Statement

Classical machine learning can develop good classification predictions; however, the current literature finds that deep learning has proven much better at making predictions by finding non-linear relations in data [47]. This transition to deep learning has been driven by the recent boom in the availability of data in conjunction with significant increases in computational power and density [21]. Deep learning is a subset of machine learning which mimics the way a biological brain learns tasks and patterns. It uses a layered structure of nodes, connected by weights and biases, for which input information can pass through to generate highly accurate decisions. Within healthcare, deep learning is present in many applications, such as in imaging diagnostics, examining unstructured electronic health records using natural language processing, or training medical robotics to learn surgical operations [22, 42, 31]. The challenge with deep learning classification is that there needs to be a significant amount of data in order for a deep learning model to find generalizable patterns from a sufficiently representative sample. When only a few data examples exist, a deep learning model will learn to make very accurate predictions on the training data. It may struggle, however, to make accurate predictions on validation data, or when deployed in a real-world setting. This phenomenon is known as the *overfitting problem*. Many times, it is too expensive, too time-consuming, or not feasible to collect more data. There is also no guarantee that the data collected will be robust when compared to the real-world data during deployment, so a need exists to ensure that the data collected is enough to make a generalizable model.

There are a few methods of combating this problem of overfitting to make a network more generalizable such as model pruning, regularization parameters, and dropout [25, 56, 38]. One such method that has demonstrated excellent potential is transfer learning. Transfer learning is the method of taking a robust pre-trained deep learning network that has been trained using a large and domain-related dataset (referred to as the *source* dataset), and then modifying that network using the dataset that needs a boost (referred to as the *target* dataset) for deep learning to be effective [49]. The premise of using transfer learning for deep learning classification is that datasets that come from similar domains possess shared features. A deep learning model may have difficulty learning these features on a small dataset. However, a similar dataset that has more data may be able to train a model to learn these features or features that are very similar to those in a smaller dataset. By first training on the large dataset, the network will learn weights and biases that are closer to what the final network will learn. After training on the larger, similar dataset, the final network will only then need to slightly adjust the weights and biases to fit the smaller dataset. In contrast, the standard procedure would begin training a new network with randomly assigned weights and biases from scratch. A classic, well-developed use of transfer learning is within the image classification domain [65]. There are many well labelled and large image datasets available online, such as ImageNet [13]. These large datasets can be used to make pre-trained deep learning models, which learn features in image recognition that have common image characteristics, such as edges, lines, and shapes. New models then don't need to learn these features and can focus on learning features specific to the smaller dataset. One example of this was a model created by Krishnadas et al. [37] which used transfer learning to detect malaria in segmented red blood cell images. Due to a lack of data, they used a ResNet50 [32] network trained on the ImageNet dataset to obtain an accuracy of 92%.

There are some considerations that need to be made when using transfer learning. When the source dataset is not chosen correctly, the retrained model may perform worse after training on the larger dataset when compared to simply training on a network with randomly initialized weights. This is known as "Negative Transfer" [49]. Fundamentally, for the success of transfer learning there needs to be a way to quantify the similarity of the two datasets. Dissimilar datasets do not have shared features and thus result in transfer learning which can be inferior to starting with random parameters. This thesis will focus its work on quantifying the correct dataset for use in transfer learning within time series classification problems, specifically time-series signals, for which there is no large robust online dataset available, akin to the ImageNet dataset for image recognition. On the Keras deep learning model repository, there are thirty-three different deep learning model architectures, all of which have been pre-trained [10]. These models can be downloaded and retrained by applying transfer learning principles. In many cases, it may not be feasible to train these models from scratch due to the size of the model and a lack of data to train. Without knowing how similar the pre-trained dataset and the target dataset are, it is unclear how much transfer learning is affecting training. Therefore, measuring the dataset similarity between the source dataset used to pre-train the model, and the target dataset, can provide insight on much of an effect transfer learning had on training accuracy.

1.2 Proposed Solution

To ensure that a deep learning researcher and/or developer doesn't waste time searching for datasets and empirically testing each of their suitability (or unsuitability), a metric for measuring the similarity between two time series datasets is proposed. Measuring the similarity of two time series datasets is a non-trivial task, due to features of different signals not necessarily aligning at the same time. By encoding signals into a feature space, measuring the similarity between those encodings is much more efficient, and this allows for measuring the similarity between two datasets. All signals are pre-processed so that they have the same signal length and frequency. Next, the signals from the source dataset are encoded using an autoencoder. This autoencoder learns to encode each time series signal into a feature embedding space. Next, a distance metric measures the average distance between the two average encoded datasets, Where low distances mean two datasets are similar. Also, a machine learning classifier is used as a discriminator, with low levels of discriminator accuracy meaning that the two datasets are similar.

The purpose of this thesis is to use a similarity metric to determine if a source dataset is suitable for transfer learning on a target dataset. Therefore, this similarity method was tested to find a correlation between dataset similarity and a boost in performance from using transfer learning.

While the focus of this thesis was to define a metric to determine the suitability of datasets for transfer learning, the demonstration of the concept was intended to be shown on a phone application called Back2Play. The purpose of the Back2Play app was to objectively predict/classify the stage of recovery for youth having experienced a concussion. For this work, the data used in this app was movement accelerometry data collected using an Apple Watch. Data was labelled for concussion stages. The goal was to use a deep learning algorithm to classify and predict the stage of concussion recovery, based on movement data collected. Due to challenges with data collection, there was a limited number of participants in the study. Only eighteen participants had data collected for use in the study and the data collected was skewed towards the final stages of recovery. As a result, a more generalizable model based on human activity recognition (HAR) publicly available datasets is presented in this thesis. There are a variety of open-source datasets used to perform human activity recognition classification, which may come from a similar domain to the Back2Play data. The similarity metrics proposed in this thesis were used to measure the similarity between the Back2Play data, and other human activity recognition datasets for potential application of transfer learning.

1.3 Thesis Contributions

This thesis presents two important contributions to the field of deep learning for time series classification: i) a novel way of quantifying the similarity between two time series datasets, and ii) a predictive approach to estimating the performance of transfer learning datasets.

Quantifying the similarity between two time series datasets: The method of using an autoencoder to first turn time series data into a series of feature embeddings, then compare the datasets by using a discriminator is presented. This thesis covers different discriminator methods that are used to discriminate the differences between embeddings, as well as examine different distance measurements as an alternative to using a discriminator.

Predictive performance of transfer learning dataset: A correlation analysis was performed to determine if these autoencoder methods of measuring dataset similarity can predict transfer learning performance. Different metrics for measuring how much performance boost can be achieved by using transfer learning are presented. These metrics were then used to evaluate the similarity metric's ability to predict how much extra deep learning performance can be obtained by using transfer learning between two datasets.

1.4 Thesis Organization

The thesis is structured as follows:

Chapter 2 presents a literature review and gap analysis on relevant research in the field of transfer learning and similarity metrics for time series datasets;

Chapter 3 presents relevant background information on deep learning and transfer learning algorithms;

Chapter 4 presents an overview of the similarity metric algorithm and describes the theory behind it;

Chapter 5 presents how to use the algorithm, as well as the experimental procedure to show how the metric was evaluated;

Chapter 6 presents the experimental validation of the similarity metric; and,

Chapter 7 discusses the findings within the experimental validation, limitations, conclusions, and future directions

Chapter 2

Literature Review

In this chapter, a comprehensive literature review on current transfer learning methods for time series classification will be presented following the guidelines for a scoping literature review in software engineering [36]. The study is summarized in three stages: 1) planning 2) conducting and 3) analyzing the review. These three stages are examined in Sections 2.1, 2.2 and 2.3, respectively.

2.1 Planning the Review

The specific research questions and rationale are presented and the search strategy is explained with search terms, exclusion criteria, and extracted metrics.

2.1.1 Research Questions

The purpose of this review was to survey the current state of transfer learning on real-time series data and evaluate their usefulness in healthcare applications. To help guide the review, a set of research questions was developed. These research questions, as well as their guiding motivations, are described below.

- RQ1: What transfer learning methods could be used on time series data in healthcare? The purpose of this question is to filter out which types of transfer learning are applicable to time series data.
- RQ2: What metrics are used to compare these models to non-transfer learning counterparts, and how does the performance compare? The motivation behind this question is to identify what parameters are used to evaluate the effectiveness of transfer learning, and how can we determine if a transfer learning approach is an improvement on a non-transfer learning approach.
- RQ3: What ways are being used in literature to measure similarity between the source and target datasets, pertaining to time-series datasets? The similar a source and target dataset are, the greater chance of a boost in performance of a transfer learning model will be. Therefore, defining a similarity metric is important in finding a potential performance boost, before transfer learning is attempted.

2.1.2 Search Strategy

In this review, the types of models that are of interest include classification of real-time series data using transfer learning or classification of medical/healthcare systems using transfer learning. Three different databases were searched, including IEEE Xplore, PubMed, and Science Direct. To examine the state of modern literature on the topic, studies were limited to after 2010. If a study came up in the search with a citation to another relevant paper, then that paper was manually searched. Each database was given the same strings to be searched for keywords that appeared either in the document title or abstract. For a study to be considered, the term “transfer learning” must appear in either the document title or abstract as a single string. For example, “transfer” cannot appear in one area of the abstract, whereas “learning appears in another. The following search terms were used:

- Transfer Learning AND Time Series
- Transfer Learning AND Accelerometer
- Transfer Learning AND Healthcare
- Transfer Learning AND Similarity

The following metrics were extracted from each of the papers if applicable, to aid in the paper selection process

- Model Class - What is the general architecture of the model used for knowledge transfer?
- Evaluation Metric - How was the model evaluated?
- Score - How well did the model score against the Evaluation Metric?
- Model Class of Competing Model - If the study compared the transfer learning model to another model, was the transfer learning model compared to a non-transfer learning model of the same architecture, or a completely different model?
- Increase in Score - How well did the transfer model do in comparison to the competing model?

Database	**Search Query: Transfer Learning AND**	**Result**
IEEE	Time Series	119
IEEE	Accelerometer	22
IEEE	Healthcare	61
IEEE	Similarity	61
PubMed	Time Series	25
PubMed	Accelerometer	3
PubMed	Healthcare	4
PubMed	Similarity	3
ScienceDirect	Time Series	25
ScienceDirect	Accelerometer	3
ScienceDirect	Healthcare	4
ScienceDirect	Similarity	0
Total		330

Table 2.1: Database Search Results

2.1.3 Exclusion Criteria

When searching for papers on transfer learning on time series data, only papers that used time-series signal data for supervised classification problems were considered for the review. Duplicate papers were removed. Papers were selected if they compare their transfer learning model to either a model with similar or equivalent deep learning architecture, but without transfer learning, or be compared to another state-of-the-art machine learning technique. Papers were also included if the focus of the study was on the process of transfer learning used.

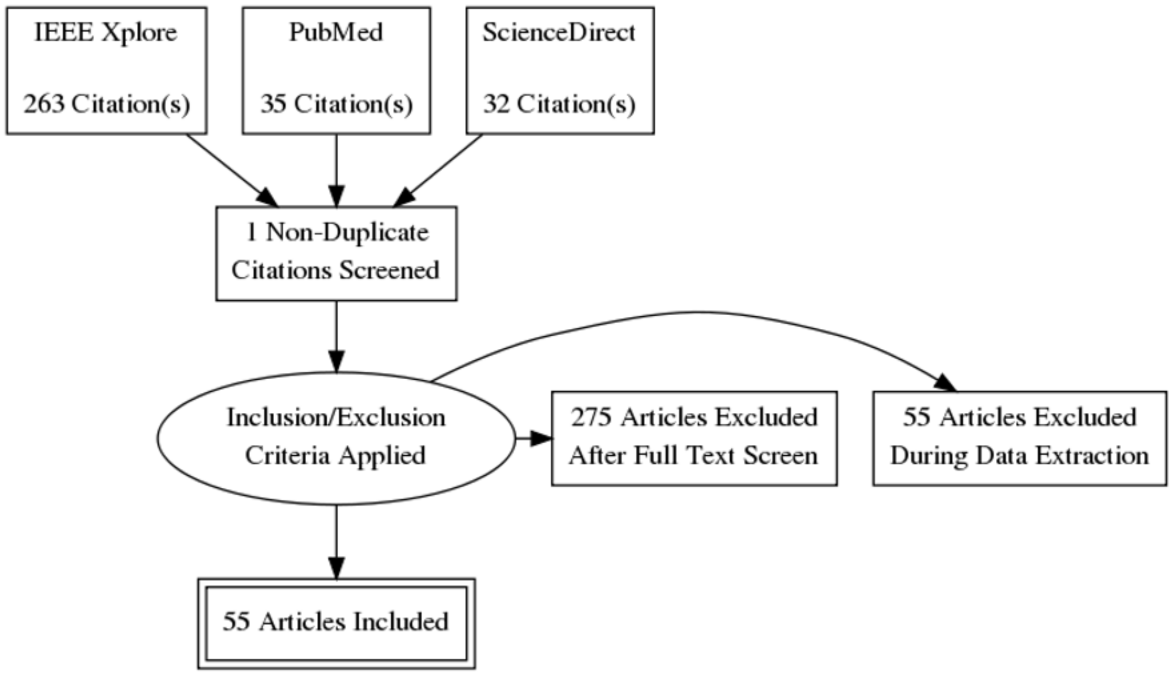

IEEE Xplore
263 Citation(s)
PubMed
35 Citation(s)
ScienceDirect
32 Citation(s)
1 Non-Duplicate Citations Screened
Inclusion/Exclusion Criteria Applied
275 Articles Excluded After Full Text Screen
55 Articles Excluded During Data Extraction
55 Articles Included

Figure 2.1: Results of screening papers for the review

2.2 Analysis

2.2.1 RQ1: What transfer learning methods could be used on time series data?

Nineteen of the fifty-five papers that were collected used transfer learning to aid in increasing the performance of a time series classification (TSC) problem. The architecture of these papers is sorted in table 2.2. Of these papers, fourteen of them used a Convolutional Neural Network (CNN) as the preferred network architecture to perform deep transfer learning. CNNs have earned their reputation as a popular neural network architecture from their use in image processing tasks; this is because CNNs are very good at feature extracting [69]. Many deep transfer learning models for TSC have source features learned using a CNN. Once the source network has learned the features, then the weights are transferred to a target network. The target network will also have a classifier at the end of the network that uses the learned features as its input. These classifiers can be a variety of traditional machine learning classifiers: SVMs [52, 53, 51], or regular Deep Neural Networks (DNN), consisting of Fully Connected (FC) layers [39, 48, 54, 9, 12, 18, 70, 19]. To help with temporal relations in the data, some networks that employ CNNs also combine Recurrent Neural Networks (RNN), such as Long Short Term Memory (LSTM), where temporal relations are learned prior to [54] or after [5, 43] CNN feature extraction. RNNs take the output of the previous time step as well as the input of the current time step to make predictions. This propagates historical information about earlier time steps throughout the network, so these early features have an influence on the current time

Architecture	**Count**
CNN	14
LSTM	3
Other	2

Table 2.2: Count of different model architectures presented for time series classification

step [8]. An LSTM is an RNN which allows for early information in longer time-series data to not vanish and still have influence dynamically. One of the problems with using LSTMs for transfer learning, specifically for TSC, is that there aren't any publicly available pre-trained networks that implement an LSTM limited public data sources to create general models [23]. Because of this, LSTM deep transfer learning is implemented differently from a CNN model. Dridi et al. [41] showed that transfer learning on an LSTM increased the LSTM's performance and training speed if the source and target data are taken from a similar domain where the source and target classes are similar. Two individuals were tracked to predict if they would be occupying their room, based on time-series sensor data. A network was trained on individual A, and the parameters learned were transferred to a target model, which was trained to predict if individual B would occupy their room. Wang et al [64] demonstrated a method where transfer learning on a joint CNN-LSTM architecture would increase performance, as knowledge was transferred from both the LSTM and the CNN [43]. Other models used included multilayer perceptron networks and some variants, such as extreme learning machines [14].

2.2.2 RQ2: What metrics are used to compare models that use transfer learning to non-transfer learning counterparts, and how does the performance compare?

In the forty-six papers reviewed that examine the application of transfer learning, metrics were used to evaluate the performance of each model, however not all studies looked at how each transfer learning model compared to that of a non-transfer learning model. There was no established or consistent metric to compare to non-transfer learning models. However, this review found that the choice of metric was dictated by model complexity, the effectiveness of other non-transfer learning models, or for unlisted reasons.

For performance, the two main methods of determining the effectiveness of these models were: i) compare the transfer learning model performance to that of a deep neural network model with the same architecture, but without the transfer learning; or, ii) compare the transfer learning model to that of another state of the art machine learning classifier performing the same task.

Comparison Type	**Count**
Same Architecture	24
New Network	15
No Comparison	7

Table 2.3: Tabulation of deep transfer learning model comparisons

Thirty-nine of the forty-six (85%) papers using transfer learning compared accuracy between a deep transfer learning model and a competitive non-transfer learning model. The architecture for comparison models is displayed in table 2.3. When comparing the two models, thirty-four of the studies showed some improvement in their

Model Metric	**Count**
Accuracy	25
AUC	8
F1	5
Kappa	1
Unreported	7

Table 2.4: Tabulation of deep transfer learning evaluation metrics

transfer learning model when compared to another competing network. Of the studies measuring accuracy as their performance metric, most studies show an increase between 0-10% accuracy. Six of the twenty-four studies that measured accuracy showed that there was negative transfer between the source and target models. This meant that transfer learning had a negative effect on the overall performance of the model [26]. As noted in section 1.1, negative transfer can happen when the source and target domains are not similar enough for knowledge from the source domain to aid in the target task [49]. If the primary purpose of transfer learning isn't to increase classification performance, but instead to increase computational speed, then transfer learning can still be a successful option. Saduok et al. [53] showed that even though applying transfer learning had a 5% decrease in the model's classification accuracy, the transfer learning model was praised for its simplicity and reduction in computation time since a CNN was not needed to be trained from scratch [53]. Table 2.4 shows the model performance metrics used to evaluate models. Twenty-five studies used accuracy as the primary evaluation metric. Other metrics that have been used to measure the effectiveness of transfer learning are AUC and F-Score, which were used by eight and five studies selected for this review.

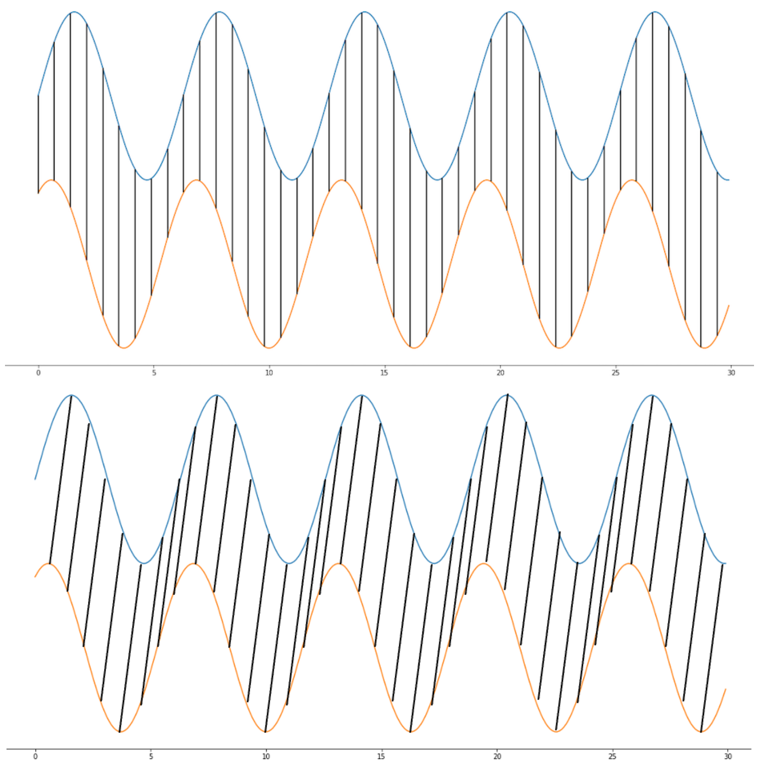


Figure 2.2: Amplitude of two signals using a lock-step method (Top) and an elastic method (Bottom)

2.3 Quantifying Similarity Between Source and Target Datasets

The similarity of the source and target datasets is an important concept in transfer learning. If the source dataset has similar features to the target dataset, then a deep network classifier trained on the source data will learn to extract many of the same features as the target dataset, which will lead to an increased performance via transfer learning [23]. On the other hand, if the two datasets are from different domains, or have no shared features between them, then the increased performance that the network receives may be minor, negligible, or negative. Therefore, when selecting a source dataset for transfer learning, some form of dataset similarity analysis should be performed to quantify the increase in performance expected by transfer learning. Due to the nature of time-series data, finding a similarity metric between the source and target dataset is not trivial. However, steps have been taken to find a metric that can quantify the difference or similarity between the source and target dataset.

The first step in quantifying the similarity between the source and target dataset of time series data is to find a way to measure the similarity between two signals. These methods fall into two categories [44]: lock-step, and elastic. In addition, the use of embeddings is included as another method of quantifying similarity.

Lock-Step: The first is a lock-step method, where both signals are compared at the same point in time. This class of approaches includes Euclidean distance, where the distance is measured as the vertical distance between two signals overlapping in time [28]. This metric excels when events happen at the same time in both signals, such as comparing the amplitudes of two ECG signals on the same person but on different body parts. This is shown in figure 2.2, where the amplitude of two signals is compared at the same time step.

Elastic: When the signal varies in both amplitude and phase, then an elastic approach is desired. Elastic measures for comparing time-series signals measure the distance between the two signals but allow for signal deformation to better match each other via a one to many or a many to one approach [16]. The most common elastic measure for comparing the similarity of two signals is dynamic time warping (DTW) [44, 20, 23, 30, 33]. This method aims to find the distance between like features in each signal, such as peaks and troughs, providing a more flexible implementation of signal similarity. This means that when events in time are not synced, a similarity measurement between the two signals can be made. A representation of DTW can be shown in figure 2.2. Along with DTW, there are other elastic measures of time series similarity, such as longest common subsequence (LCS) and all common subsequences (ACS) [20]. It is important to note that DTW is not a real distance metric, since it does not satisfy the triangle inequality [57, 16, 44]. Fawaz et al. [23] explains a method of using DTW Barycenter Averaging (DBA) to find the mean time series of a class within a dataset. The study then uses the DTW algorithm to measure the intra-dataset similarity by finding the distance of each class within a dataset, as well as inter-dataset similarity, by measuring the DTW distance between all classes in one dataset and all classes in another, then finding the minimum of those distances. Surprisingly, there is a lack of testing similarity measures for evaluating inter-dataset similarity, let alone using these measures as a predictor for transfer learning performance. However, many studies have used these methods of finding the similarity between groups of signals in clustering algorithms, such as K-Means [28, 63, 67]. In the future, more work should be done on using time-series signal to signal similarity measures as a way to quantify inter-dataset similarity, as well as predict the performance of transfer learning based on this similarity.

Embedding: Although not currently found in the literature for time series classification using transfer learning, another method of finding similarity between datasets is derived from the embeddings of a trained neural network. Embeddings are found by training a neural network on a dataset and extracting the weights and biases of the network [34]. The dissimilarity between two neural networks trained from different datasets is proportional to the distance between the set of embeddings of each example. Embeddings allow a simpler distance calculation, such as Euclidean distance, rather than DTW. Mathisen et al. [46] has demonstrated this principle by using a traditional ANN to create embeddings on many datasets in the UCI data repository, as well as using CNNs to create embeddings on images, and using those embeddings to find the similarity of two images. As Mathisen et al. points out, an interesting avenue of research would be to create embeddings on time series data and use those embeddings to measure similarity between signals, as well as the inter-dataset similarity between datasets containing time series, since deep neural network architectures have been shown to accuracy classify time series data, and therefore can lean these embeddings. To measure the distance between these embeddings, many different distance measurements can be used. Euclidean distance is the baseline, however more complex distance measurements can also be used. Wasserstein Distance [40], or the Maximum Mean Discrepancy [17], which measures the dissimilarity of probability measures between the source and target domains, are some of the measurements used to quantify the difference between these embeddings, as it pertains to the whole dataset. These metrics can be used as optimizers, to selectively choose which information to transfer to the target domain, which is known as conditional transfer learning. One study that employed conditional transfer learning used two different networks as the source and domain models. The source model was a ResNet CNN model trained on ImageNet data, and the target model was an LSTM-ResNet model trained on images of faces for GH pituitary tumours. The target model produced a feature vector of a face, and the source model made a support vector. These vectors were sent to an attention kernel, which, with help from an LSTM to learn a connection between the source and target vectors, produces an output [15].

2.4 Literature Review Summary

Transfer Learning is used as a method to increase the performance of deep learning models for time series classification. A variety of architectures are used, depending on the classification problem. The performance of a transfer learning model can be evaluated with respect to a non-transfer learning model by comparing various evaluation metrics between the two models. The current primary metric used to compare the two models is empirical accuracy. There is a lack of literature that attempts to predict the performance gained from the use of transfer learning. All papers in this review that focused on an application of transfer learning, did so without any insight on the feasibility of transfer learning based on the compatibility of the two datasets. This presents a clear gap in the literature that this research seeks to address.

Chapter 3

Similarity Metric Design

In this chapter, the design of the similarity metric is presented. Discussed here will be the theory on how to generate encodings using an autoencoder. Two methods of measuring similarity between the source and target dataset encodings are also discussed.

3.1 Motivation

Transfer learning involves using a supplementary (source) dataset as a booster to increase the generalizability of a model when the original dataset a model is meant to classify (target) is too small. The model is first pre-trained on the source dataset, then the weights are transferred to the target model, which is then trained on the target dataset. To ensure that the target model isn't confused, features should be shared across both datasets. When shared features exist, the target model does not need to worry about learning these features, as they have already been learned from the source data. However, if there are little to no shared features, then the model may be confused about how to use features that don't exist in the target data to make classifications.

Therefore, there is a need to ensure that a source dataset is chosen that has sufficient enough shared features with the target dataset. To do this, the similarity between the two datasets must be considered. When dealing with time series datasets, there are considerations that must be made when comparing two signals, let alone two datasets. A digital signal can be represented as a time series vector, so an elementary comparison between two signals can be made by measuring the difference in amplitude of the signal at each time step. This is a viable comparison; however many assumptions must be made. First, each signal must be of the same length and sampling frequency, so that the two time series vectors are the same shape. More importantly, the features from each signal must align perfectly in time. For example, if comparing two time series accelerometer signals of a person walking forward, the first step must occur at precisely the same time. A phase-shifted signal that would otherwise be very similar to an original signal would have scored as being very dissimilar. There are methods of correcting for this misalignment, such as Dynamic Time Warping, however, these methods are still not perfect, as the initial and final time points are still meant to be in alignment [50] .

This thesis attempts to solve the time-variance problem of comparing time-series signals by transforming each signal into a space where two signals can be easily compared with each other. This allows each signal from a dataset to be grouped together and compared against signals from another dataset. To do this, an autoencoder is used to turn a raw signal into a feature vector, independent of time.

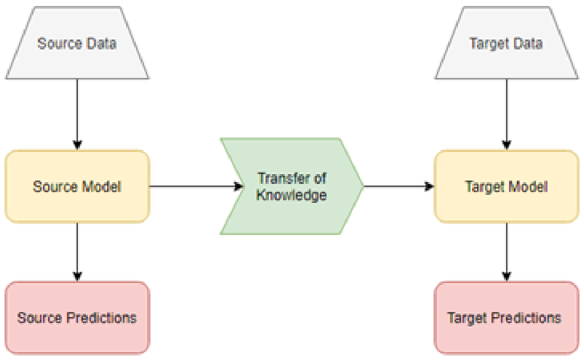


Figure 3.1: An overview of knowledge transfer from one deep learning model to another.

3.2 Theoretical Background

3.2.1 Transfer Learning

Transfer learning is a technique to improve the performance of a machine learning model, by leveraging information learned in one domain and applying it to another. Pan and Yang [49] gives a good definition of transfer learning. A domain D is represented by the two following parts: a feature space χ, and a probability distribution P(X), where X belongs to χ. A task T also contains two components: a label space y, which are ground truth labels to X, and a prediction function f(x), which makes a prediction on what the ground truth of an example in X. With these terms defined, Pan and Yang give the following formal definition of transfer learning. Given a source domain D_s and learning task T_s, a target domain D_t and learning task T_t, transfer learning aims to help improve the learning of the target predictive function $f_t(x)$ in D_t using the knowledge in D_s and T_s, where D_s does not equal D_t, or Ts does not equal T_t.

Figure 3.1 provides a visual description of the concept of transfer learning. Source data is used to train a model and make accurate predictions on source labels. Any relation between the source domain and the target domain, whether implicit or explicit, can be transferred to the target model. This extra knowledge can allow the target model to learn faster, as well as give the model a greater data source to learn more information, which can result in greater prediction performance. The focus of this thesis will be on a method of transfer learning known as deep transfer learning, and how it can be used in time series classification for healthcare. Deep transfer learning falls under the category of transfer learning known as inductive transfer learning. Both the source data and the target data are labelled [6]. In classification problems, this is generally the case. Deep transfer learning, as defined by Tan et al. [59], is the following: Given a transfer learning task defined by $\langle D_s, T_s, D_t, T_t, F_t(\cdot)\rangle$. It is a deep transfer learning task where $F_t(\cdot)$ is a non-linear function that is modelled in a deep neural network. A deep neural network is a class of machine learning model architectures with several layers of neurons that are meant to simulate the learning structure of the brain. The input layer collects data, the output layer makes a prediction on the label of the data, and the hidden layers perform levels of abstraction, to learn complex features and patterns within the data. The layers closer to the input of the neural network perform low-level feature extraction, such as edges in an image [66]. As information is fed forward to the later layers, more specific features of the classification task are learned. Deep transfer learning applies this concept, by having a source network learn on source data, then transferring the parameters learned to the target network. If the source and target network are applied to similar datasets, then the source model will have learned low-level features from the source data which can be the same as the low-level features on the target data.

There are two types of deep transfer learning used in classification problems: Fine-tuning the weights and freezing the weights. The most common method is fine-tuning, or model weight transfer. The method of fine-tuning the network is as follows. First, a source network is trained on source data to create a model meant to classify the source data. Then, that model's parameters, such as weights and biases, are transferred to the target model. The target model may replace the output classification layers to better fit the target task. The model is then retrained. the source parameters are set as the initial parameters, and training is done using the target data. The benefit of this is if the source data contains similar low-level features to the target data, then the source model can extract features from this data and use them to help classify the target data. This is a good technique when there is limited target data. The source data can give the network a boost to learn these features, which can prevent the model from overfitting.

3.3 One Dimensional Convolutional Neural Networks

As deep learning has developed, specialized network architectures have been developed to perform specific tasks. One of these networks is the convolutional neural network (CNN). The CNN is an architecture wildly used in computer vision and image recognition problems [55], due to its ability to recognize patterns that repeat throughout an image, as well as lower the computational complexity, due to weights and biases being reused throughout the image [35]. An artificial neural network (Or a multi-layer perceptron network) contains an input layer and weights that connect each feature from the input layer to hidden layers. Each node in a hidden layer has weights connecting the node to each node in the previous and next layer. This is problematic when images are being classified, since images create a large number of features, as a single pixel represents a feature. Each pixel is treated as independent, which prevents the network from reusing learned features across the image. CNNs rectify these problems by passing a convolution filter across the image. The nodes of the CNN are the filters that are passed through the image, and the values in the kernel are the weights that are learned via backpropagation. A sample 2D CNN, described in [35], describes the process of classifying an image:

1. A linear convolution is performed between the input image and the first filter. The shape of the filter and the number of filters are predetermined by the network designer. The shape of the new image is $(k_x - 1)(k_y - 1))$, where k_x and k_y represent the filter shape. The number of filters used determines the number of channels in the new feature map.

2. The feature map is passed through an activation function.

3. The feature map is passed through a pooling layer. This reduces the shape of the feature map by finding the maximum or average of a group of features in the feature map. For example, if a pooling layer of 2x2 is used, then the shape dimensions of the feature map will reduce by half.

4. The first three steps are repeated, providing depth to the network. A flatten layer is used to transform the final feature map into a single vector.

5. Dense layers are used to finish the network, similar to a vanilla neural network. The final layer does the classification.

These networks are trained by backpropagating the classification error through the network and are optimized using a selected variant of gradient descent [35]. A similar algorithm has been developed for one-dimensional data: one-dimensional CNNs (1DCNN). They operate on the same principles as the 2DCNN, however, instead of sliding a 2D filter across an image, a 1D filter is slid across a time-series signal. The 1DCNN receives many of the same benefits as the 2DCNN, such as the ability to reuse features learned throughout a signal. For example, if a 1DCNN is used in human activity recognition, a filter may learn a specific walking pattern early in the signal. That filter can be used to detect the same walking pattern later on in the signal or in other signals.

3.4 Autoencoders

Autoencoders are a type of neural network which first compresses information, then reconstructs the compressed information as accurately as possible [7]. The network consists of two separate networks, an encoder and a decoder. The encoder has information passed through, which reduces the dimensionality of the information. At the end of the encoder is the layer with the least number of nodes, called the bottleneck layer. At the bottleneck layer, the information is as compressed as possible. The second part of the network is the decoder. The decoder takes encoded information and attempts to recreate the original information[1]. The size of each layer increases from the bottleneck layer until the layer shape is the same shape as the original information. Training an autoencoder is a supervised learning problem. Data is compressed

[1]Conceptually this is no different than a digital signal encoder/decoder, or the process used to construct/deconstruct Morse code.

by the encoder, reconstructed by the decoder, then compared to the original data using mean squared error (MSE) loss. This loss is backpropagated through the network until the reconstruction loss reaches a minimum.

Autoencoders can be used on a variety of data types, including time-series data. To use a time-series autoencoder, the data must first be transformed from a time series signal to a feature vector when encoding the data. This can be done by using a one-dimensional convolutional neural network (1DCNN). Convolution layers and pooling layers are used to extract features from the signals, as well as reduce the dimensionality of the data. In this thesis, the data used have short repetition cycles that are well suited for use in 1DCNNs, such as accelerometer signals and ECG recordings. if the signals are longer and non-repetitive, such as the price of a stock over time or fluctuations in wireless signal strength, a recurrent layer, such as an LSTM or GRU should be used after the convolutional layers, to retain information that happened earlier in the signal [68]. The signals are then flattened, and dense layers are added to reduce the size of the signal until the bottleneck layer is reached. To decode the signal, a mirror of the encoder is used. Dense layers that increase in size are used to increase the dimensionality of the signal until the signal is reshaped to the shape of the signal in the encoder before flattening. Then, the signal is transformed back to its original shape by using depooling layers and deconvolution layers to undo the transformations the encoder made. The output layer of the decoder has the same shape as the input layer of the encoder.

When the network is trained, the autoencoder can transform signals from a time-dependent signal to a feature vector and back, while retaining key information. This allows for signal reconstruction. This is key for similarity measurements of two signals since the signals are now in a space where they are easily comparable, due to them now being time-invariant.

3.5 Dataset Transformations

This section covers how to use the autoencoder to transform a source and target dataset into a space where the two datasets can be compared.

3.5.1 Training the Autoencoder

The first step is to train the autoencoder on the source data. Training on the source data allows for greater autoencoder accuracy, due to the amount of data available being greater than the target dataset. The target dataset is split 90% training and 10% validation. Training is done by feeding the raw signals through the autoencoder, which are reconstructed after the decoder. The loss of the autoencoder compares each time step of the reconstructed signal to the original signal. Training is done over 100 epochs, however, validation accuracy was monitored, and early stopping was implemented so that overtraining doesn't occur. This process is repeated five times, each time with a different random initialization of model parameters. The autoencoder with the highest accuracy is selected as the autoencoder for the trial.

3.5.2 Encoding the Datasets

After the autoencoder was trained, the source and target datasets were encoded. This was done by removing the decoder portion of the autoencoder. Now, the output of the network is an encoded feature vector of the signal, with a shape dependant on the size of the bottleneck layer, rather than a signal being recreated at the end of the network. The source and target datasets were passed through the new encoder separately, and two datasets encoded datasets were returned. The two datasets had then been transformed from a set of time dependant signals to feature vectors.

3.6 Similarity Measurements

Once the two datasets were encoded, the similarity between the two encoded datasets was measured. Since the source dataset was used to train the autoencoder, source encodings retained much of the original signal information. The target dataset will have embeddings which may be more varied than the source dataset. When two datasets were similar and features were shared, the target dataset occupied a similar space in the feature vector space as the source dataset. When the target dataset was dissimilar to the source dataset, then the target signals were not encoded accurately and did not occupy the same feature vector space.

It is the goal of this thesis to provide a quantitative measure of similarity between the source and target dataset. Many methods are possible, but they can be split into two categories: ”modelled” and ”learned”. This naming convention was referenced from Mathisen et al. [45], which looked at finding a way to measure intra-dataset similarity between classes in a dataset. A ”modelled” similarity metric is a measure pre-designed, which requires no machine learning. An example of this is the Euclidean distance between the source and target centroid embeddings. This metric is defined as

$$Dist = \sqrt{x_1^2 + x_2^2 ... x_n^2}. \tag{3.6.1}$$

The second metric type is a "learned" metric. Instead of using a defined measure, a machine learning model can be used to optimize the similarity, based on the data. Although more complex, this type of metric may be more powerful, since it optimizes itself to fit the data best. In this thesis, the "learned" method of measuring dataset similarity will be the use of a discriminator. The discriminator is a classifier that looks to classify the source dataset from the target dataset. If the source and target dataset are similar, then the discriminator has a difficult time distinguishing the dataset encodings from each other. Otherwise, if the datasets are dissimilar from each other, then the discriminator will be more accurate in classifying the two datasets.

Chapter 4

Experimental and Implementation

4.1 Datasets

There are six datasets used in this thesis. One is used as a source dataset, and five are used as target datasets. Each dataset consists of time series signals and come labeled for supervised learning.

4.1.1 UCI HAR

The University of California, Irvine Human Activity Recognition (UCI HAR) [60] dataset is an open source online dataset used for classifying six different human activities. These activities are walking, walking upstairs, walking downstairs, sitting, standing and laying down. Thirty volunteers between 19-48 years of age were used to collect data on each activity. Each signal is taken from the accelerometer and gyroscope in a Samsung Galaxy S II smartphone, worn on the participant's waist. The data collection frequency is 50 Hz, and each signal is 2.56 seconds long, resulting in 128 readings per window of signal. The data from this dataset comes pre-filtered to remove any gravitational components. Only accelerometer readings were used to make any classifications. Each acceleration signal comes with three signals: one for each of the three dimensions of movement. The total number of signals within the dataset is 10299.

This dataset was chosen to be the source dataset for a variety of reasons. First, it is well used in literature as an easy human activity recognition dataset to generate HAR models for. A simple search on IEEE Xplore for "UCI HAR" returns 57 results, all pertaining to using the dataset for HAR classification. This dataset was chosen because of its low number of classes, and minimal pre-processing needed. The classes are balanced, so training a source model on this dataset can generate a robust classifier for both static and dynamic activities. The preprocessing steps performed on this dataset are clearly outlined [61], so it is easy to match the target dataset's properties to the UCI HAR datasets, such as signal length and frequency.

4.1.2 UCI DSA

The UCI Daily Sports and Activity [4] is another HAR dataset found on the UCI repository. This dataset consists of nineteen different activities, including all six activities found in the UCI HAR dataset, as well as more dynamic activities, such as playing basketball. Eight participants between the ages of twenty and thirty performed the activities for five minutes, with each signal from the activity divided into five second signals. Each participant wore five sensing units: one on their torso, one on the left and right arm, and one on the left and right leg. Each of those sensing units contained a three-axis accelerometer, which measured acceleration in three dimensions of movement. Each sensor captured readings at 25 Hz, resulting in 125 readings per window. In this thesis, each different sensor module is treated as its own dataset, meaning that in the UCI DSA dataset, five separate datasets are extracted.

4.1.3 WISDM

The Wireless Sensor Data Mining (WISDM) HAR dataset [61] is another HAR dataset which has 18 different classes, however only five are used in this thesis: walking, sitting, jogging, standing and walking up or down stairs. This dataset contains four sensors to record time series signals. Fifty subjects are used to create signals that are three minutes long and are recorded at a frequency of 20 Hz. Two sensors are accelerometers, which record acceleration on the participant's smartphone in their pocket, as well as on a smartwatch worn on their wrist. The other two sensors come from the same smartphone and smartwatch; however the gyroscopic recording was used instead. These extra sensors were included to provide a different type of recorded data for the similarity metric.

4.1.4 ECG Datasets

All aforementioned datasets are within the HAR domain and are expected to therefore have some similarity to the UCI HAR dataset. To include datasets that are dissimilar, two ECG datasets are also used in this thesis. Both datasets can be found as part of the PhysioNet competition [27] [2]. The first is the WFDB Georgia dataset, which contains over 10,000 ECG recordings. These recordings are between 5 and 10 seconds long with a sampling frequency of 500 Hz. The second database is the Physikalisch-Technische Bundesanstalt (PTB) dataset. This dataset contains samples of varying length, between 10 seconds to 120 seconds, and are sampled at either 500 Hz or 1000 Hz. Each recording contains twelve signals, one for each of the twelve leads used in an ECG. to ensure that these recordings fit the similarity metric input, only the first three leads are kept. This may lead to less accuracy; however the work of this thesis is not to achieve high accuracy models, but to see if transfer learning can improve these models based on the similarity between the source and target dataset.

4.2 Deep Learning Models

For this research, transfer learning was done between deep learning models. These deep learning models were used to classify time series data. Each deep learning model for every classification experiment was created to have the same architecture. This was done to ensure that each model was compared consistently. There is also a practical motivation to why the same architecture was used for each model. When searching for a dataset to use as a source for a transfer learning application, machine learning engineers want to limit time cost of developing models. Optimizing hyper-parameters of a source model represents a large time cost to engineers. Unrefined models can still give good decent accuracy if the data is clean enough. Again, the purpose of this work is not to optimize the classification, but instead this work seeks to quantify improvement that transfer learning can provide a priori, thus offering a more convenient metric than empirical test and re-test.

The architecture for the deep learning models used in this thesis are less complex compared to today's state of the art deep learning models. A graphical representation of the architecture can be found in figure 4.1. Each deep learning model contains a signal input layer, followed by a 1D convolution layer with 32 filters. MaxPooling is

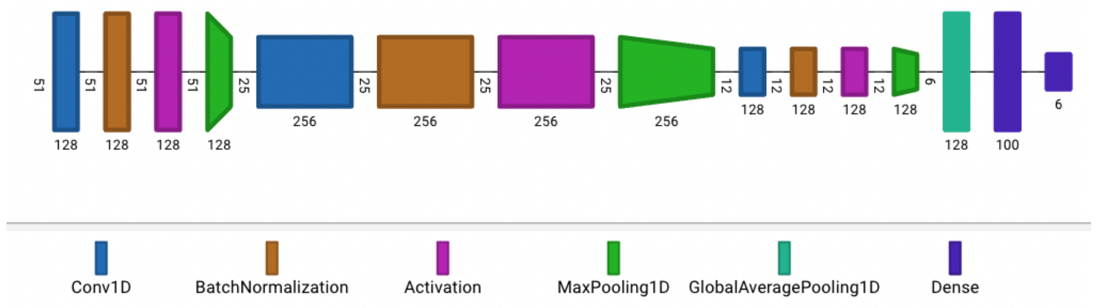


Figure 4.1: 1DCNN architecture for classifying signals. The input to the classifier is a three channel signal of 51 time steps

then used to reduce the size of the signal four-fold, and dropout is used as a measure to prevent overfitting. These three layers are repeated again, with the convolution layer now having 64 filters. The feature map is then flattened to a vector of 128 neurons. A dense layer of 50 neurons is the last hidden layer, which provides some more non-linear feature mapping before a final dense layer makes a prediction on the class. The output layer's shape is changed to fit the number of classes in a particular dataset.

4.3 Similarity Metrics

Two similarity metrics are discussed in this thesis: one modeled similarity metric, and one learned similarity metric.

4.3.1 Manhattan Distance

The Manhattan distance (or L1 distance) is the distance between two points where the shortest path available is calculated based on the space's bases vectors, rather than a straight line. It is called the Manhattan distance because to move between two places in Manhattan, it is impossible to drive in a straight line from one point to the other. A car must drive in a grid like pattern north or south, then east or west. This distance measurement metric is used rather than Euclidean distance because of the curse of dimensionality [1]. The greater the number of dimensions, the less meaningful Euclidean distance becomes. The formula for Manhattan distance is shown in equation 4.3.1:

$$d(I_1, I_2) = \sum_{P} |I_1^P - I_2^P| \tag{4.3.1}$$

Manhattan distance was chosen as the preferred metric for measuring distance in the encoding space due to the curse of dimensionality [1]. As the dimensions of a space increases, the data within that space becomes sparser. Distance metrics with lower norm values have greater contrast between points for sparse data. Therefore, Manhattan distance has a norm metric of one, therefore it was chosen over alternative distance metrics, including Euclidean distance, which as a norm metric of two.

If the two datasets have similar features, their embeddings will be clustered near each other. Conversely, if the two datasets are dissimilar, then the target dataset should be clustered away from the source dataset. Therefore, the modeled similarity metric is the Manhattan distance between the centroids of the source and target dataset. These centroids are calculated by finding the average of each feature for the entire dataset.

4.3.2 Discriminator

The discriminator approach uses a machine learning model to determine the similarity between two datasets. This concept was derived from the Generative Adversarial Network literature [3] where a generator and discriminator are used to compete with

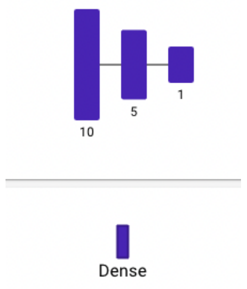


Figure 4.2: The discriminator architecture

each other; the generator tries to create data that statically matches the domain, and the discriminator tries to detect the "faked" data. For the purposes of this thesis, the discriminator concept has been applied to quantify the difference between two datasets as though the target was "generated" and the source was used to train the "discriminator." The learned discriminator needs to quantify how different are the two encoded datasets (source and target). For this quantification, a simple neural network was employed and all data points were labelled in the embedding space as either belonging to source or target dataset. The neural network was taught to distinguish the two datasets. If the network has difficulty determining difference, then the two datasets are similar. If the network easily distinguishes the two datasets, then the two datasets are dissimilar. A low complexity network was empirically chosen to improve performance. It is believed that the lower complexity acts to increase the quantization binning size for classification which was suitable for this complexity of data. This approach provided improved range in the overall accuracy between similar and dissimilar models. Instead of a distance between two values, the similarity metric is an accuracy score of how well the discriminator was able to discriminate the source data from the target data.

The similarity is scored between 0.50 and 1.00:

0.50 means the discriminator was randomly guessing, unable to discriminate the two datasets from each other; and,

1.00 means that the discriminator was able to predict which dataset an encoding belonged to 100% of the time.

Accuracy was validated using data unseen during training and based on a percent split using an 85-15 train test split.

4.4 Evaluating Transfer Learning Performance

Transfer Learning performance is a difficult parameter to measure. Two different metrics are used throughout this thesis to give a better understanding if there is a positive boost in performance from using transfer learning:

1. overall accuracy improvement, and
2. area between the transfer learning curves.

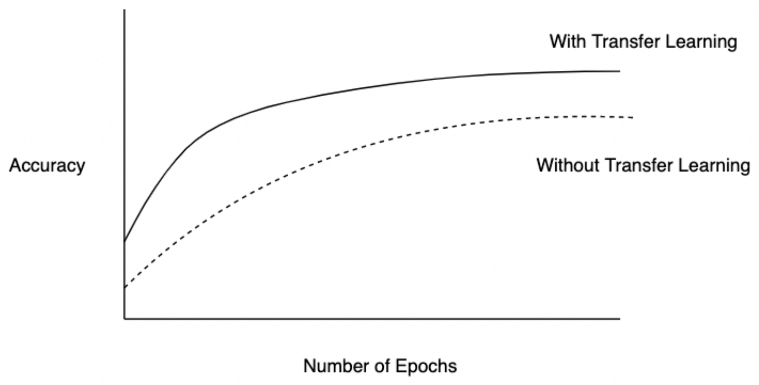
4.4.1 Overall Accuracy Improvement

The first metric that is evaluated is overall accuracy improvement. To measure this, the validation accuracy of the model that included transfer learning was compared to the validation accuracy of the model without the use of transfer learning. The best accuracy of each model was used, so if the highest validation accuracies for each model happened at different epochs, those validation accuracies would still be used, rather than choosing the training accuracy at the last epoch or at matching epochs.

The author's logic behind this choice was based on the accepted training termination, called "early stopping". Early stopping will terminate training when accuracy criteria are met rather than continuing to train until (for example) 100 epochs. However, these models did not terminate their training with early stopping so that we could observe the same range of training cycles for each dataset.

Sometimes, a dataset may not need transfer learning, but can still see a benefit to using the technique. There are three indicators that transfer learning has had a positive effect on learning:

1. The overall accuracy of the model has increased. This is captured in the first performance metric.
2. The first epoch accuracy of the model has increased.
3. The rate at which accuracy is increased over the first few epochs is higher.



Transfer learning has more benefits than just increasing the validation accuracy on a test set. It also makes the model more generalizable when deployed, allowing the model to have a better chance of accurately classifying unseen patterns in data.

Therefore, it is important to note when transfer learning is successful, even when the overall transfer learning accuracy isn't much higher than a model without transfer learning.

4.4.2 Area Between the Curves

To capture the overall effect of transfer learning, a new metric of measuring transfer learning performance is introduced here. This new metric is the area between the transfer learning curves (ABtlC). When transfer learning has a great positive effect on learning, the area between these two curves increases. To simplify this calculation, rather than take the area between the curves, the difference in accuracy at each epoch is calculated and summed over the entire training period.

$$ABtlC = \sum_{n_{epochs}} (a_{tl} - a_{notl}) \tag{4.4.1}$$

Equation 4.4.1 is another metric which evaluates transfer learning performance when the amount of overall accuracy boost a model can get is negligible. This metric is preferred when the original non-transfer learning model has already achieved high accuracy.

4.5 Pre-Processing

In this thesis, three experiments were run to determine the validity of similarity metrics. First, the similarity between the UCI HAR dataset and itself with noise was performed.

4.5.1 Signal Processing

Since the UCI HAR dataset came with a lot of processing done on it, it remained generally unprocessed throughout the thesis. One processing step that was not done on the UCI HAR dataset was signal normalization. Each signal was normalized have a maximum power of 1.

$$s(t) = \frac{s(t) - min(s(t))}{max(s(t)) - min(s(t))} \tag{4.5.1}$$

Each other dataset was processed to have similar characteristics to the UCI HAR dataset. First, each signal was resampled using the SciPy resample function [62], to a frequency of 50 Hz, which uses a Fourier method of resampling. Next, each signal was windowed to a signal length of 2.56 seconds, and 128 readings per window of signal. This was done to match the shape of the target signals to that of the autoencoder. Signal segmentation was done using a fixed-width sliding window with 50% overlap. The signals were filtered using a Butterworth high pass filter with a cut-off of 0.3 Hz to remove gravitational noise.

Transfer learning is a method useful for increasing accuracy and generalizability in datasets with low data samples. Therefore, to simulate smaller dataset sizes, the target datasets were randomly sampled to 500 samples after windowing. This represented approximately 5% of the size of the UCI HAR dataset. This was done to give the non-transfer learning model a harder time finding generalizable features, lower accuracies, and more susceptible to overfitting. This way, a boost from transfer learning is expected to be easier to quantify. The datasets were reduced at random, and five-fold validation was used. For training and validation, a 70-30 train test split was used.

4.5.2 UCI vs UCI + Noise

It was important in the early stages of this thesis to make sure that meaningful similarity measurements were made before transfer learning is considered. To do this, the similarity metrics needed to be compared to some ground truth similarity between two datasets. One way of achieving this certainty of similarity is to compare a single dataset with an altered version of itself. The more the dataset is altered, the less similar it becomes to the original dataset. By applying the similarity metric between the original and altered dataset, a prediction can be made on how altered the dataset is, and then compared to how altered the new dataset actually is.

The original dataset used in this experiment is the UCI HAR dataset. This dataset will be used as the source dataset moving forward. To create a new dataset altered from the UCI HAR dataset, a known amount of artificial noise is injected into each signal of the dataset. Eight different datasets were created by injecting different levels of noise. The amount of noise injected was determined by the desired SNR of the new signals. SNRs of each dataset were set to 40, 35, 30, 25, 20, 15, 10 and 5. To determine how much noise to inject, the signal power for each of the three accelerometer axis was calculated. The power of the noise created was calculated by dividing the noise of the original signal by the desired SNR. A noisy signal was created by generating a Gaussian noise signal, with the mean of the noise equalling the mean of the original signal. The range of amplitude of the noise equalled the square root of the noise power. This noise was then added to the original signal. This process is shown in figure 4.3

$$NoisePower = SignalPower/SNR \tag{4.5.2}$$

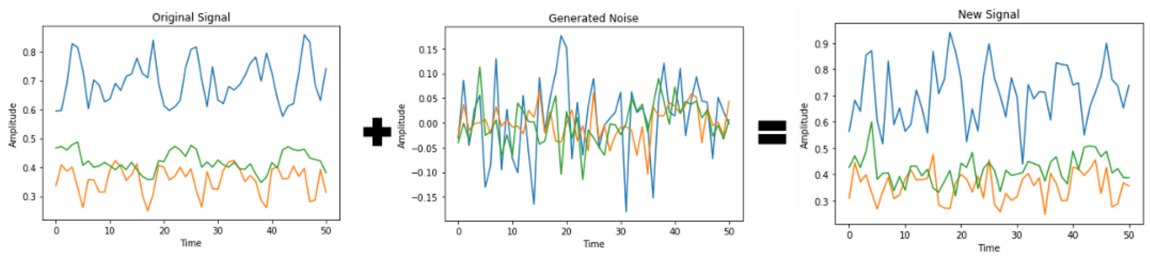


Figure 4.3: The generation of a noisy signal by adding a known amount of Gaussian noise to the original signal

$$Noise = Rand(Mean, Amplitude), \tag{4.5.3}$$

where $Mean = mean_{signal}$, $Amplitude = \sqrt{NoisePower}$.

The standard deviation of a signal is a measure of how far the amplitude of a signal deviates away from the mean of the signal. When Gaussian noise is added to a signal, the chance that the amplitude of that signal deviates from the mean becomes higher. When averaged over a large dataset, if large amounts of Gaussian noise is added to each signal, the likelihood that each signal has a greater standard deviation increases. One way to quantify this change in likelihood is to measure the standard deviation of the standard deviation of the dataset. The standard deviation of each signal in the dataset is recorded, and the standard deviation of the distribution of standard deviation is measured. Histograms of noisy datasets can be seen in figure 4.4. As the SNR of signals increases, the standard deviation of each signal tends to increase, as the total power of each signal increases due to the addition of extra noise.

Figure 4.5 shows the distribution of standard deviation of the signals shows a clear change in the dataset by changing the SNR of the signals in the datasets. As more noise is added to the dataset, specifically around the 15 SNR range, the more different

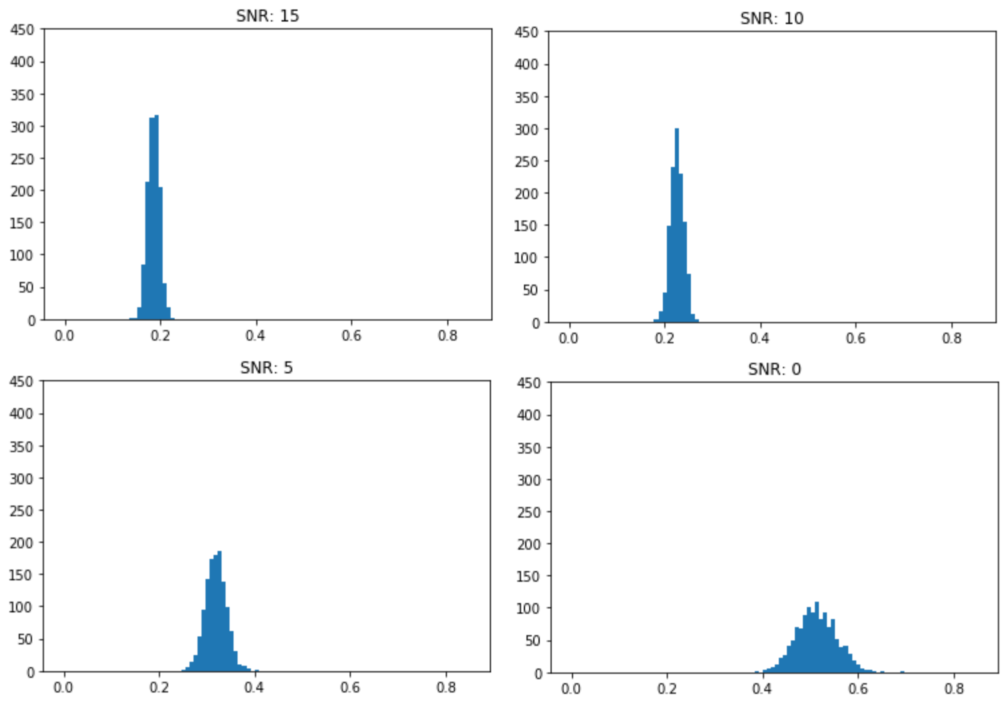


Figure 4.4: Histogram of the standard deviation of signals

the datasets become. The standard deviation of the signal is a measure of how far the signal strays away from the mean amplitude. When the noise of the signal begins to overpower the original signal, the greater the chance that the new signal begins to stray away from the original signal's mean, resulting in a greater standard deviation of the signal. A histogram of the dataset's standard deviation of signals for different SNRs for the walking class's x-axis is plotted in figure 4.5. The standard deviation of these plots stays relatively the same for SNRs between 40 to 20, but once the noise reaches past 20 SNR, a widening of the histogram can be seen. The standard deviation of these plots is displayed in figure 4.4 showing the increase in standard deviation as the SNR goes down.

To test the metric, the untouched, original UCI HAR database was used as the source dataset. Each of the newly created and altered UCI HAR datasets was used

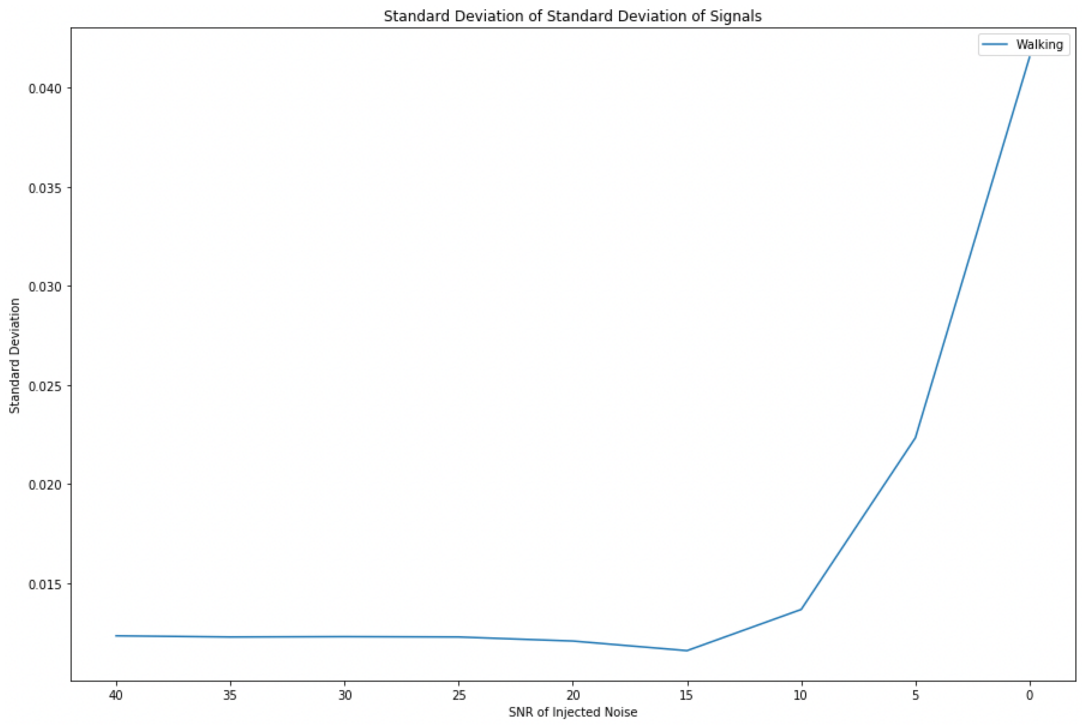


Figure 4.5: The standard deviation of the standard deviation of signals in a dataset

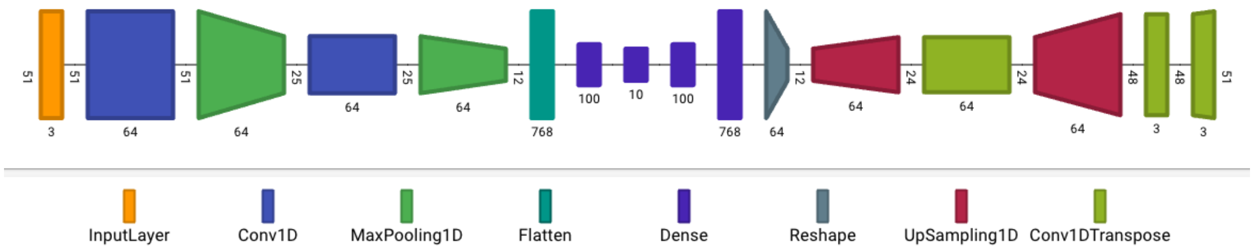


Figure 4.6: Autoencoder architecture. The entire network is used for training. All layers past the ten node bottleneck layer are a part of the decoder and were discarded after training.

as the target dataset. The source dataset was used to train the autoencoder and the source model.

4.6 Creating the Encoder

An autoencoder was trained and the encoder was extracted to evaluate dataset similarity. The architecture for the autoencoder is displayed in figure 4.6 Five autoencoders were trained, and the autoencoder with the lowest reconstruction loss was chosen as the autoencoder to use throughout the thesis. Mean squared error was the loss used for training, and the adam optimizer was used to update the weights of the autoencoder. A learning rate of 0.001 was used. The autoencoder with the lowest reconstruction loss after training had a reconstruction loss of 0.0093.

4.7 Experimental Methods

In this section, the experimental methods used to evaluate the similarity metrics are described. Three experiments were conducted. The first two experiments attempt to prove the two similarity metrics can measure the similarity between two datasets by measuring the response of injecting noise to the target dataset. The final experiment attempts that the similarity metrics can accurately predict the potential performance gains from using transfer learning.

4.7.1 Noise Injection Similarity: Same Dataset

The following steps were used to evaluate $\text{UCI}_{original}$ vs. $(\text{UCI+Noise})_{set}$:

1. An autoencoder is created and trained on the $\text{UCI}_{original}$ HAR dataset.
2. $\text{UCI}_{original}$ HAR dataset and one of the eight different noisy datasets ($(\text{UCI+Noise})_0$, $(\text{UCI+Noise})_1$, ... $(\text{UCI+Noise})_7$) are then passed through the encoded, generating two embedded datasets.
3. The similarity between the $\text{UCI}_{original}$ HAR dataset and $(\text{UCI+Noise})_{set}$ are evaluated, without evaluating transfer learning.

The similarity between the encoded datasets is measured using the two similarity metrics presented in section 4.4. The results of these similarity metrics are presented in section 5.1 as plots comparing the results of the similarity metric to the amount of noise injected. A good similarity metric should see a decrease in measured similarity as the amount of noise injected increases.

4.7.2 Noise Injection Similarity: Different Datasets

Using the same methodology as the subsection 4.7.1, the similarity metric is tested by measuring how different two datasets are when noise is injected into one. However, while the source dataset is still the UCI HAR dataset, the new target dataset is now a different time series dataset. Recall these datasets were:

- UCI Daily Sports and Activities: A human activity recognition dataset
- WISDM: A human activity recognition dataset
- WFDB GA: An ECG classification dataset

Similarly, the autoencoder is trained on the UCI HAR dataset. Noise is injected at the same levels of SNR as in the previous section, to the UCI DSA dataset (Torso), the WFDB Ga dataset, and the WISDM dataset (From the smartphone accelerometer). The autoencoder is used to generate encodings of the UCI HAR dataset and the different target datasets. The similarity of the two datasets is expected to decrease as the SNR of the noise injected dataset decreases, however, because the source and target are distinctly different datasets there should be an initial difference when no noise is injected (unlike UCI$_{original}$ vs. UCI+Noise).

4.7.3 Similarity Metric vs Transfer Learning

The similarity metric is then compared to potential performance gains by using transfer learning. An autoencoder is created and trained using the UCI HAR dataset. Then, each of the eleven other datasets used in this thesis has their similarity measured against the UCI HAR dataset. A deep learning model is then trained from scratch with random initialization on one of the target datasets. Transfer learning is then applied between the UCI HAR and the target dataset. Another deep learning model is trained on the UCI HAR dataset, and then each trained parameter is transferred to another model, which will be trained on the target dataset. The only

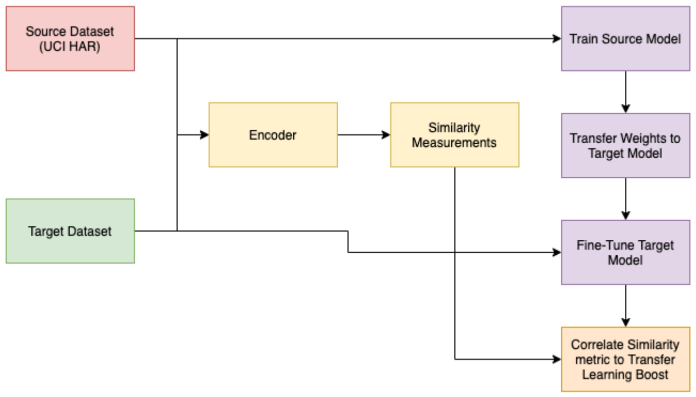


Figure 4.7: Process diagram for comparing the similarity metric to the boost in performance from transfer learning

parameters that are not transferred over are the final layer's weights, which are reset back to randomly initialized weights. The architecture for each of these models is described in section 4.2. Each target network's final layer is customized to fit the number of labels that the target dataset is trying to classify. The transfer learning network is then retrained on the target data. This is repeated for each of the other target datasets. The difference in highest validation accuracy between the transfer learning network and the non-transfer learning network, as well as the area between curves, are measured. Once transfer learning is performed, the similarity between the source dataset and the target datasets is measured using both the Manhattan distance and the discriminatory accuracy. A correlation between the similarity between the source and target dataset, and the performance boost gained from transfer learning is made. Transfer learning is most needed when there is a lack of data. The target datasets were reduced to 1000 samples each, compared to the over 9000 samples in the UCI HAR dataset. The metrics were also evaluated at dataset sizes of 800, 600, 400 and 200 examples. A process diagram is shown in figure 4.7

Chapter 5

Results

In this chapter, the experimental results of the similarity metric are presented. The chapter is structured as follows: Section 6.1 provides a breakdown of how the similarity metric performed in a controlled setting, measuring the similarity between the UCI HAR dataset and a modified version of itself. Section 6.2 is the results of measuring the similarity between two different datasets, with noise injected. Section 6.3 gives the results of measuring the similarity of the two datasets and compares it to how much transfer learning performance was measured.

5.1 Noise Injection Similarity: Same Dataset

In this section, the results of the similarity metric measuring a dataset and an altered version of itself is presented. The results of measuring the similarity using the modelled Manhattan distance metric and the learned discriminator metric are displayed in table 5.1.

SNR	**Manhattan Distance**	**Discriminator Accuracy**
40	0.0049 ±0.0001	0.51 ±0.03
35	0.0100 ±0.0002	0.51 ±0.03
30	0.0200 ±0.0003	0.51 ±0.06
25	0.039 ±0.007	0.52 ±0.04
20	0.077 ±0.007	0.53 ±0.01
15	0.15 ±0.02	0.56 ±0.08
10	0.29 ±0.03	0.61 ±0.07
5	0.48 ±0.04	0.70 ±0.06

Table 5.1: Comparison of the two similarity metrics to measure the similarity between the UCI HAR dataset and a noisy version of itself.
Note: For both metrics $\downarrow$ value means $\uparrow$ similarity.

5.2 Noise Injection Similarity: Different Datasets

This section presents the results for finding the similarity between the UCI HAR dataset and other datasets used in this thesis. Noise was injected in the same way as in the previous experiment. Table 5.2 shows the results of measuring the similarity between the UCI HAR dataset and the Daily Sports and Activities dataset, with the signals collected on the torso accelerometer. Tabular results of the other datasets can be found in 7.3

SNR	**Manhattan Distance**	**Discriminator Accuracy**
40	1.153 ±0.0	0.922 ±0.006
35	1.155 ±0.0	0.923 ±0.002
30	1.158 ±0.0	0.925 ±0.004
25	1.166 ±0.001	0.926 ±0.004
20	1.176 ±0.001	0.927 ±0.006
15	1.19 ±0.002	0.932 ±0.003
10	1.216 ±0.004	0.928 ±0.004
5	1.293 ±0.015	0.937 ±0.002

Table 5.2: Comparison of the two similarity metrics to measure the similarity between the UCI HAR dataset and a noisy version of the UCI Daily Sports and Activities

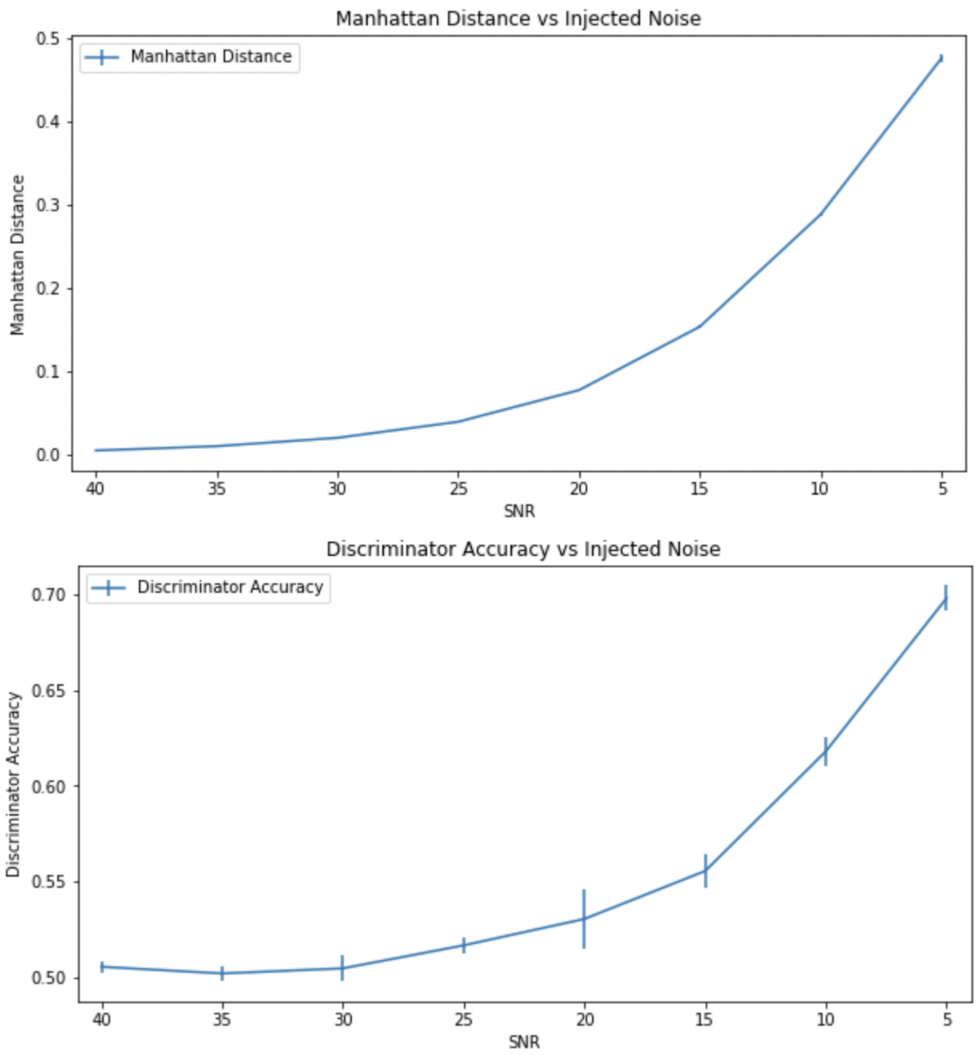


Figure 5.1: Top: The measured Manhattan distance between the UCI HAR dataset and noise-injected versions of the UCI HAR dataset. Bottom: The trained discriminator accuracy between the UCI HAR dataset and noise injected versions of the UCI HAR dataset

5.3 Dataset Similarity vs Transfer Learning

In this section, the results of measuring the correlation between the similarity metric and the performance boost given from transfer learning is presented. No noise is added to any of these datasets, however, the target datasets are modified by reducing

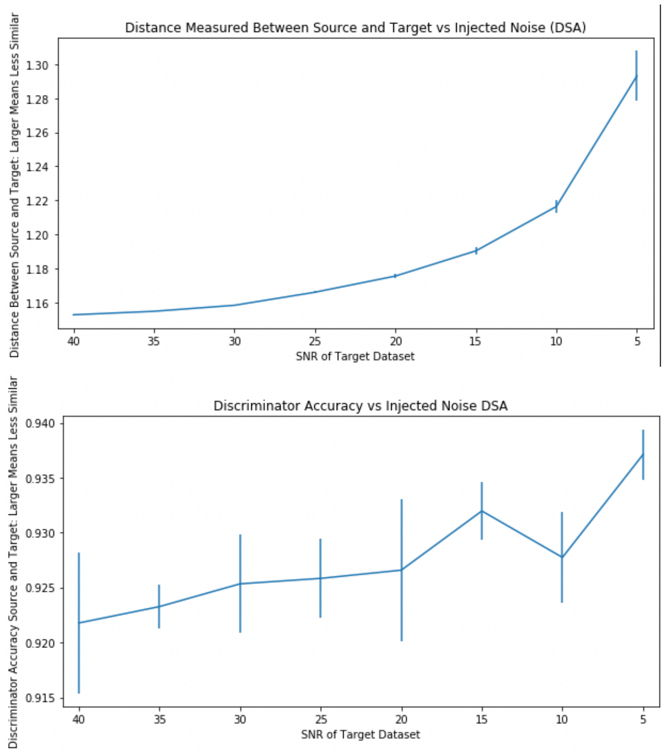
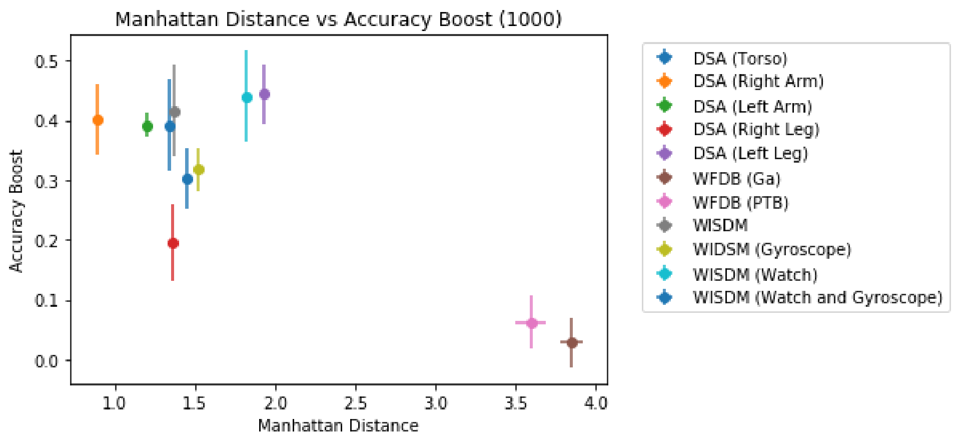


Figure 5.2: Manhattan distance and discriminator accuracy between the UCI HAR dataset and the noisy Daily Sports and Activities dataset

the number of examples. This is done to simulate the typical use case of transfer learning, where the amount of training data is insufficient to train a model on its own. One design consideration that arose is what should the size of the bottleneck layer be? Larger bottleneck layers resulted in less autoencoder training loss, which resulted in better represented encodings. Smaller bottleneck layer sizes resulted in the discriminator metric having a harder time distinguishing the source and target dataset on datasets that were very similar. This resulted in a higher variance in the similarity scores across the datasets. The distance metric also had an improvement when using a smaller bottleneck layer. Distance metrics carry well into higher dimensional space [1], so a sufficiently small bottleneck layer should be used. A 10-unit bottleneck layer was used as a compromise for these two problems. Table 5.3 presents the results of the experiment measuring dataset similarity and transfer learning performance, respectively.

Dataset	**Manhattan Distance**	**Discriminator Accuracy**	**Transfer Learning Boost**	**Area Between Curves**
DSA (Torso)	1.34±0.03	0.944±0.004	0.39±0.08	58±4
DSA (Right Arm)	0.89±0.01	0.883±0.007	0.4±0.06	53±3
DSA (Left Arm)	1.2±0.01	0.926±0.006	0.39±0.02	48±6
DSA (Right Leg)	1.362±0.003	0.976±0.004	0.2±0.06	22±7
DSA (Left Leg)	1.93±0.01	0.977±0.002	0.44±0.05	40±3
WFDB (Ga)	3.85±0.07	0.984±0.002	0.03±0.04	6±2
WFDB (PTB)	3.59±0.09	0.97±0.002	0.06±0.04	5±2
WISDM	1.373±0.002	0.86±0.01	0.42±0.08	29±7
WIDSM (Gyroscope)	1.521±0.004	0.925±0.004	0.32±0.04	15±4
WISDM (Watch)	1.821 pm 0.009	0.95 pm 0.003	0.44 pm 0.08	35 pm 7
WISDM (Watch and Gyroscope)	1.452 pm 0.004	0.884 pm 0.007	0.3 pm 0.05	23 pm 3

Table 5.3: Dataset similarity results at 1000 examples



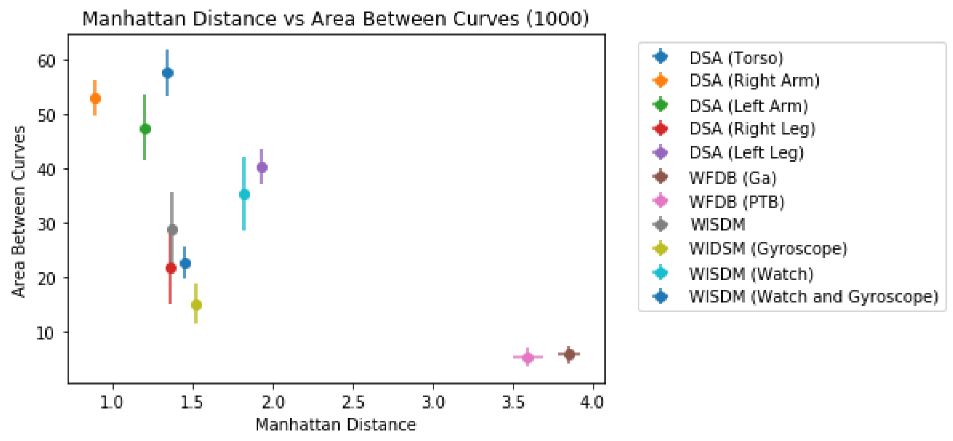


Figure 5.3: Transfer learning performance vs Measurements of Manhattan distance results with the target dataset set to 1000 examples

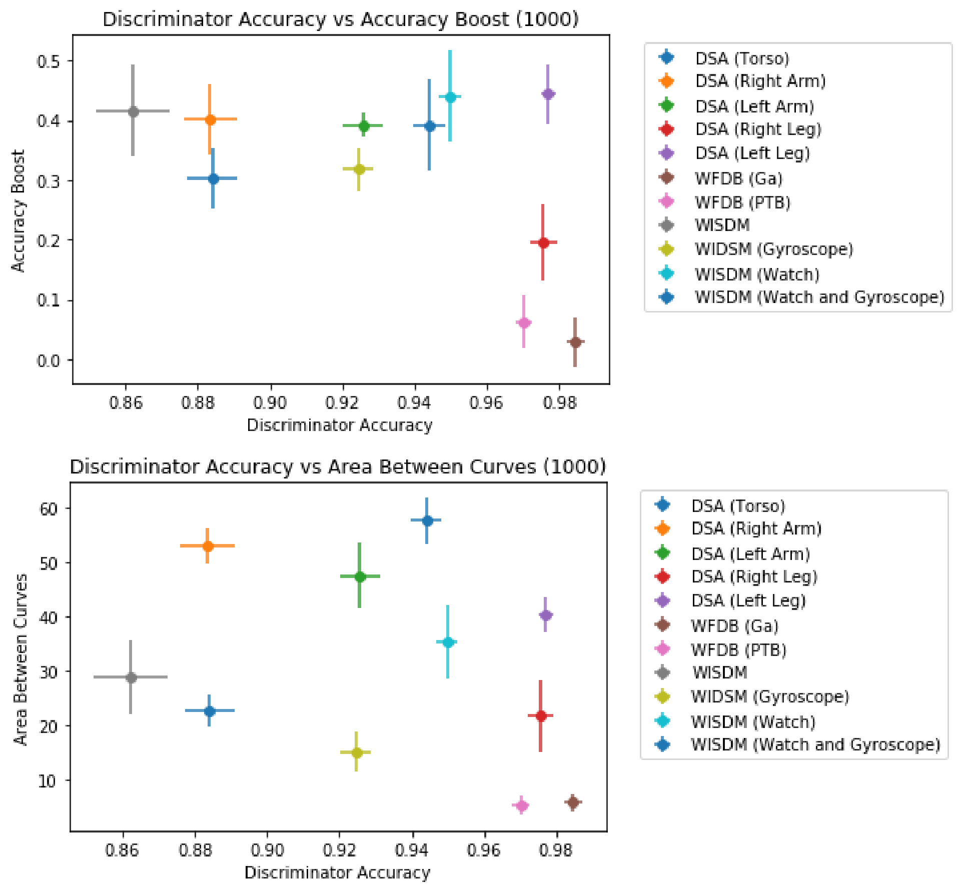


Figure 5.4: Transfer learning performance vs Discriminator accuracy results with the target dataset set to 1000 examples

Chapter 6

Discussion

6.1 Research Findings

This section of the thesis discusses the results presented earlier in this chapter.

6.1.1 Same Dataset Similarity

The results presented in section 5.1 show that both the Manhattan distance and the discriminator can be used as a metric for measuring dataset similarity. Using the noise injection method presented in section 4.7.1, it was expected that as the amount of noise injected into each signal increased, the greater the dissimilarity between the two datasets was going to be. The experimental results confirm this hypothesis. With noise injections between 40 and 25 SNR, both the Manhattan distance metric and the discriminator accuracy metric did not detect much of a difference between the two datasets. This may be due to the low amount of noise injected having little to no effect on the actual signal integrity. With SNRs of 20 and lower, both the Manhattan distance and the discriminator accuracy metrics began to detect a difference between the two datasets. The distance between the centroids of the original and noisy datasets became larger, and the discriminator had an easier time learning which dataset a data point came from.

There are some noted differences between the two metrics. When an SNR of 40 was used, the Manhattan distance metric detected no difference in the two datasets. This was the result of the autoencoder generating the same embedding for each of the two datasets. Because of this, the centroids for each dataset lay on top of each other, resulting in a distance metric near zero. Also because of this overlap in embeddings, the discriminator was unable to learn which dataset each embedding came from since each dataset is essentially the same. Therefore, the discriminator would take a random guess at which dataset the embedding came from. Since there are two datasets, the encoder would get a correct prediction around fifty percent of the time. As the data gets nosier, the difference between the two datasets becomes more apparent. The encoder is less able to generate the same encodings as the original UCI HAR dataset is. This results in a shift in the location of the embeddings in the embedding space. This shift is detected by the Manhattan distance metric since the centroid of the noisy dataset is now further away from the original dataset's centroid. This shift was the highest when the most noise was injected. Similarly, the discriminator was able to detect a shift in the two datasets as they began to differ. As the encodings shifted, a decision boundary was able to be developed separating the two datasets. This meant that the discriminator no longer was guessing at random, but learning which data came from what dataset. The best performing discriminator was when the noise injected reached 5 SNR, resulting in the highest discriminator accuracy.

This agreed with the Manhattan distance metric, as well as the expected results.

Observing the plots of each similarity metric vs SNR in figure 5.4, the discriminator metric presents less precise measurements. This is observed by the higher variance in discriminator accuracy, which is most noticeable at 20 SNR. This variability is due to the random initialization of weights in the discriminator, and the discriminator is trained for a fixed number of epochs.

6.1.2 Different Dataset Similarity

The results from section 5.1 carry over to results of section 5.1, supporting the author's assertion that the experimental results carry over to real transfer learning problems. This experiment measures the same metrics as the same dataset similarity experiment, however, the target dataset (where noise will be injected) comes from different datasets, not the UCI HAR dataset. All three datasets examined exhibited the same exponential relationship between the distance measured between the UCI HAR dataset embeddings and the new embeddings from the target dataset with respect to noise. However, since the two datasets are not the same, the highest SNR example did not have a distance metric close to zero, since the encoder is not generating the same embeddings for both datasets. The UCI DSA and the WISDM dataset had the lowest distance measured between their embeddings and the UCI HAR embeddings, followed by the WFDB Ga dataset. Intuitively, this makes sense because the UCI DSA dataset and the WISDM datasets each have data recorded on accelerometers, within the human activity recognition domain. This is the same domain as the source UCI HAR dataset, recorded from accelerometers. The furthest dataset from the UCI

HAR dataset is the WFDB Ga dataset; since this dataset is time series ECGs for cardiac abnormalities, rather than accelerometer data for human activity recognition, it is expected that this dataset is the least similar to the UCI HAR dataset.

The discriminator metric also measured a decrease in dataset similarity when the amount of noise injected increased for all datasets, however, like in the first experiment, the results were nosier. The curve in figure 5.2 does not match the discriminator curve from figure 5.1. This is because the two datasets when the noise injected is no longer the same. This resulted in discriminator accuracies greater than random guessing. The result is a much more gradual increase in the discriminator accuracy curve.

Where the discriminator model fails is in cross-domain similarity measurements when the noise is increased.

6.1.3 Dataset Similarity vs Transfer Learning

The purpose of this thesis is to construct a metric to quantify the performance of transfer learning by comparing the source and target datasets prior to conducting the transfer learning. Figure 5.3 shows that the Manhattan distance metric is negatively correlated with both the area between transfer learning curves and transfer learning boost metrics. The error in the distance metric is low, suggesting that even with random dataset shrinking, the metric will still be valid. The Daily Sports and Activity (Right Arm) dataset had a measured Manhattan distance of 0.89, which was the shortest distance measured between all the datasets. This dataset also had the fourth largest boost measured boost from transfer learning, with an accuracy boost of 0.4, and the second highest area between transfer learning curves, with a value of 53.

Therefore, if a machine learning engineer were to attempt transfer learning to improve the training results of the Daily Sports and Activity (Right Arm) dataset, selecting the UCI HAR dataset would be a good option. Conversely, there was a large distance between the centroids of the UCI HAR dataset and both the ECG datasets. This was expected, given the dissimilarity of the domains. The UCI HAR dataset, therefore, was not a suitable source for trying to improve the performance of both ECG datasets.

It is interesting to see that the discriminator accuracy was not a viable metric when the dataset size was 1000 examples. Although the two ECG datasets that both had dissimilar domains to the UCI HAR datasets still exhibited dissimilarity by having a high discriminator accuracy, other datasets that would be suitable for transfer learning also had high discriminator accuracies. The discriminator accuracy metric also exhibited a much larger uncertainty in dataset similarity. The Manhattan distance metric did not experience this uncertainty; therefore, this uncertainty must be due to the discriminator learning different decision boundaries between the two datasets every time. This is believed to be the result of how the standard random initialization of weights and biases affect the discriminator.

6.1.4 Effect of Target Dataset Size

The target dataset size in this thesis ranged from 1000 examples down to 200 examples. While the ultimate goal for transfer learning is to increase accuracy and generalizability, the transfer learning accuracy boost performance metric turned out to be unreliable when the number of examples was very low. This is for two reasons. First, with the low number of training examples, there isn't a large enough sample size to cover enough of the dataset distribution; this meant there was an opportunity for examples that are not captured in the training set to be captured in the test set. This would mean that transfer learning may have a huge impact on network performance. There also may have been other trials where a good sample of data is captured in the training and test set which would allow for training and testing accuracy on the non-transfer learning models, reducing the potential performance gains of transfer learning. This dichotomy of settings creates a large variance in the potential performance of transfer learning boost measured. Second, is that at very low dataset sizes, the accuracy of the model can vary drastically from epoch to epoch. Only 15% of the data was used for testing. This means that for a 200 example dataset, only 30 examples were used for testing. If only one of those examples changes from a true prediction to a false prediction, the accuracy of the model goes down by 3.3%. This can be shown by looking at the learning curves of models trained on 200 example datasets. They have many large steps in testing accuracy, rather than the smooth curves expected when training models. The area between transfer learning curve metric tended to be more reliable at lower dataset sizes. This is due to any spikes in the accuracy of either the transfer learning model or non-transfer learning model having less of an effect on the overall metric. Figure 5.3 shows how the area between transfer learning curves metric demonstrates transfer learning performance. The transfer learning model clearly started learning faster and reached its overall accuracy quicker than the non-transfer learning model, even though the non-transfer learning model had similar accuracy. Transfer learning in this case would still be useful to speed up training time and create a more generalizable model.

6.2 Application in Back2Play

The Manhattan distance metric was used to measure the similarity between the Back2Play study and two different human activity recognition datasets: The UCI HAR dataset, and the UCI Daily Sports dataset, where data was collected from a sensor worn on the left arm. The original Back2Pay dataset had six different return to action (RTA) classifications, however, these classifications were binned into three classes. Stages one and two were binned into a stage, stages three and four were binned into a stage, and stages five and six were binned into a stage. The model used to train the Back2Play dataset was the same as the other classification models used in this thesis. A non-transfer learning model was trained on the Back2Play data and the results of this model are shown in table 1. The non-transfer learning model achieved a final accuracy of forty-eight percent, with classifications skewed towards the final two classes. To improve this classification accuracy, transfer learning between the two human activity recognition datasets was performed. Before applying transfer learning, the measure of similarity between the datasets was measured. A Manhattan distance of 7.51 was measured between the Back2Play dataset and the UCI HAR dataset, and a Manhattan distance of 5.42 was measured between the Back2Play dataset and the UCI Daily Sports dataset. These similarity measures predict that a negligible performance boost would be achieved if transfer learning was performed. When transfer learning was performed, the total performance gains from transfer learning were negligible and can be observed in table 1.

Changes made in the Back2Play data may give both better non-transfer learning performance, as well as greater dataset similarity, resulting in greater transfer learning performance. The Back2Play data consisted of accelerometer signals worn by a participant in the study throughout the day. The accelerometer on the Apple Watch measured movement throughout the day, regardless of what activity was being performed by the participant. Concussion recovery stages were labelled independent of activity performed, so activities such as walking, running and sitting were all given the same label. This made it difficult for a deep learning model to detect the different recovery stages. Recording participant motion when only when specific activities are performed, such as walking, will allow the model to look for differences in only walking patterns related to concussion stages, rather than many different activities.

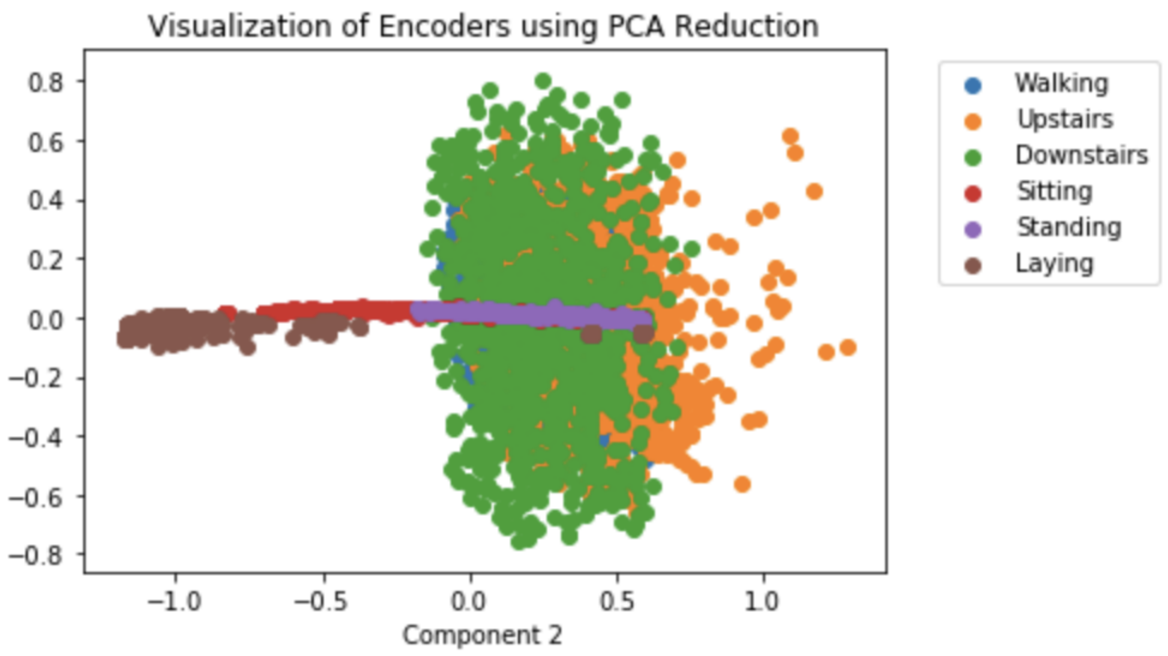
Given the limited data and stage skewness, the Back2Play dataset was not suitable for the demonstration of the metric. However, as more data is collected another method that may increase the Back2Play model accuracy is by using a "wide and deep" approach [11]. The deep approach would use the model architecture described in this thesis to perform feature extraction. Features are also extracted from the signals using traditional methods, such as signal mean, signal maximum, and signal standard deviation. These features are used as the inputs to a shallow neural network, known as the wide network. The output of these networks is then concatenated and used to perform classification. By using both deep feature extraction and statistical feature extraction, more information is provided to the classifier to recognize patterns in the data.

6.3 Limitations

The metrics both exhibit some limitations. First, the performance of deep learning models depends greatly on hyperparameters. Hyperparameters are the features of a neural network that can be changed, such as the number of hidden layers, the size of each hidden layer, the use of regulation tools such as dropout etc. While hyperparameter tuning is a common practice in deep learning, an interesting problem arises when performing transfer learning. Since both the source network and the target network need to have the same hyperparameters, a comparison between a transfer learning model and a model without transfer becomes difficult, due to the hyperparameters not being optimized for the source network.

A limitation with both similarity metrics is that they do rely on the autoencoder generating a good encoding. The autoencoder in this thesis was trained five different times, and the model with the lowest training loss was used. Due to random initialization of weights, the loss of these models varied from 0.0028 to 0.00095. Using principal component analysis (PCA), the dimensions of the encoder can be reduced for visualization. Figure 6.1 shows two different trained encoders with different training losses. Encoders with poor reconstruction accuracy created different encodings on both the source and target data compared to encoders with good reconstruction accuracy. Having an accurate encoding of the source model is crucial to making sure that the similarity metrics pick up the difference between the source and target datasets. To this point, hyperparameters may need to be tuned in order to create a good source encoding.

An autoencoder can only accept a signal with one signal length, as the number of timesteps of the signal needs to match the number of input nodes in the first layer. Therefore, both the source signal length and the target signal length need to be the same size. To do this, either the source or target signals need to be manually adjusted to match each other size. Although windowing the signals may lead to greater model performance, both datasets needing to be the same size may cause one



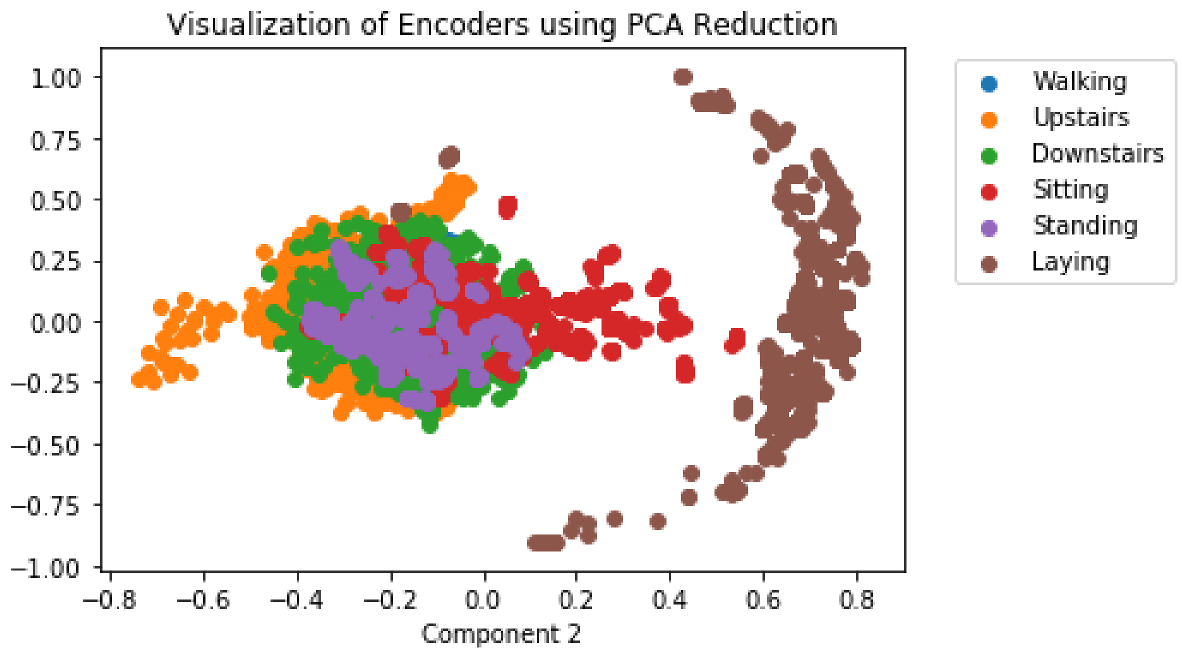


Figure 6.1: UCI HAR reconstruction on a high loss autoencoder (Top) and a low loss autoencoder (Bottom)

of the dataset's signal lengths to be windowed too small for accurate classification. One future adjustment may be to include customizable input and output layers that can accept any target signal size, before feeding the signal into the autoencoder trained on the source dataset. However, that would lead to the encoder trying to construct an encoding of features of the target signal, and not the raw signal itself. This would lead to the similarity metric comparing two different types of data, leading to higher distance measurements and greater discriminator accuracies.

Chapter 7

Conclusion and Future Work

7.1 Summary

In this chapter, a summary of the contributions made by this thesis are summarized and future research directions are discussed.

7.2 Summary of Contributions

The proposed research was to develop a metric that would quantify dataset similarity to determine if a source dataset was a suitable (or most suitable) candidate for transfer learning for the purpose of performing deep learning on time series classification. Due to the difficulty of comparing two time-series datasets, an autoencoder was trained and used to transform the data into a time-independent feature space. The autoencoder used a one-dimensional convolutional neural network architecture and was trained on source data. Two metrics were used to compare the differences between the two datasets in this space, and the metrics were then evaluated to predict the potential gains in accuracy and performance when transfer learning is applied to the two datasets.

7.2.1 Similarity Between two Datasets

This research successfully shows that each metric described in this thesis can quantify the difference between two time-series datasets. To measure this similarity, the UCI HAR dataset was modified in a controlled way, and compared with an original version of the UCI HAR dataset. To modify the dataset, known amounts of noise were added to the dataset. As more noise was added to the dataset, the Manhattan distance between the two datasets got larger, and the discriminator was able to more accurately learn which dataset each embedding came from. This proved that the encoder was preserving features from the signals, and the metrics were able to distinguish when sufficient noise added altered the entire dataset. Furthermore, the research also showed similar results when now comparing the UCI HAR dataset to other time-series datasets. A limitation on the discriminator was that at higher levels of dissimilarity, the discriminator has an upper limit on how much of a difference it can detect between the two datasets.

7.2.2 Predicting Transfer Learning Performance

This research successfully shows that the metrics described in this thesis can also be used to predict how much transfer learning performance can be obtained. The Manhattan distance metric between the two datasets proved to be the most successful, both in terms of a correlation between transfer learning performance, as well as in the stability of the metric at different dataset sizes. It also had the lowest variance between trials. The discriminator model was less successful in predicting the exact amount of transfer learning performance gained, but an overall negative trend was still present. Target datasets that were largely dissimilar to the source datasets did not provide much transfer learning boost, which correlated with a high discriminator accuracy. Target datasets that were largely similar to the source datasets did provide a much greater transfer learning boost, which correlated with a low discriminator accuracy.

7.3 Future Work

The metrics may be further refined by examining different dataset similarity methods. There are many similarity metrics that measure the difference between two probability distributions, such as KL divergence [58] and Bhattacharyya Distance [29]. Metrics like these are particularly sensitive to outliers and should be measuring the differences between two normal distributions. If a transformation was performed to turn the encoded data to a probability distribution, then use any of these metrics to measure the similarity, it would be interesting to see if these distance measurements could predict transfer learning better.

Another future avenue of research is to continue to develop metrics that predict the performance boost from transfer learning. Because transfer learning is usually done on such small datasets, it is difficult to develop a baseline model upon which transfer learning can improve. Due to random initialization of weights, one model may learn generalized patterns and accurately predict the test data, whereas another model may try to overfit whichever class has the most data to obtain the highest accuracy on the training data. The result is a large discrepancy between what a non-transfer learning model may be able to achieve in terms of accuracy. This creates large differences in the total amount of accuracy gained when using transfer learning. The area between curves metric does give a more stable result when comparing two models, but continued development on creating a more consistent non-transfer learning model should be explored. Finally, different types of transfer learning should also be explored to see if the metrics still hold. When two datasets come from different domains, substituting pure weight transfer for domain adaptation [24] has been shown to reduce the chances of negative transfer.

SNR	**Manhattan Distance**	**Discriminator Accuracy**
40	1.097 ±0.0	0.889 ±0.007
35	1.104 ±0.0	0.888 ±0.005
30	1.118 ±0.0	0.888 ±0.005
25	1.147 ±0.0	0.884 ±0.003
20	1.209 ±0.001	0.891 ±0.007
15	1.32 ±0.002	0.894 ±0.005
10	1.5 ±0.003	0.893 ±0.006
5	1.782 ±0.006	0.897 ±0.004

Table 1: Comparison of the two similarity metrics to measure the similarity between the UCI HAR dataset and a noisy version of the WIDSDM dataset

SNR	**Manhattan Distance**	**Discriminator Accuracy**
40	3.724 ±0.0	0.985 ±0.001
35	3.725 ±0.001	0.988 ±0.001
30	3.726 ±0.003	0.986 ±0.001
25	3.731 ±0.003	0.987 ±0.002
20	3.741 ±0.006	0.987 ±0.002
15	3.786 ±0.007	0.986 ±0.001
10	3.887 ±0.014	0.986 ±0.002
5	4.087 ±0.029	0.985 ±0.002

Table 2: Comparison of the two similarity metrics to measure the similarity between the UCI HAR dataset and a noisy version of the WFDB GA dataset

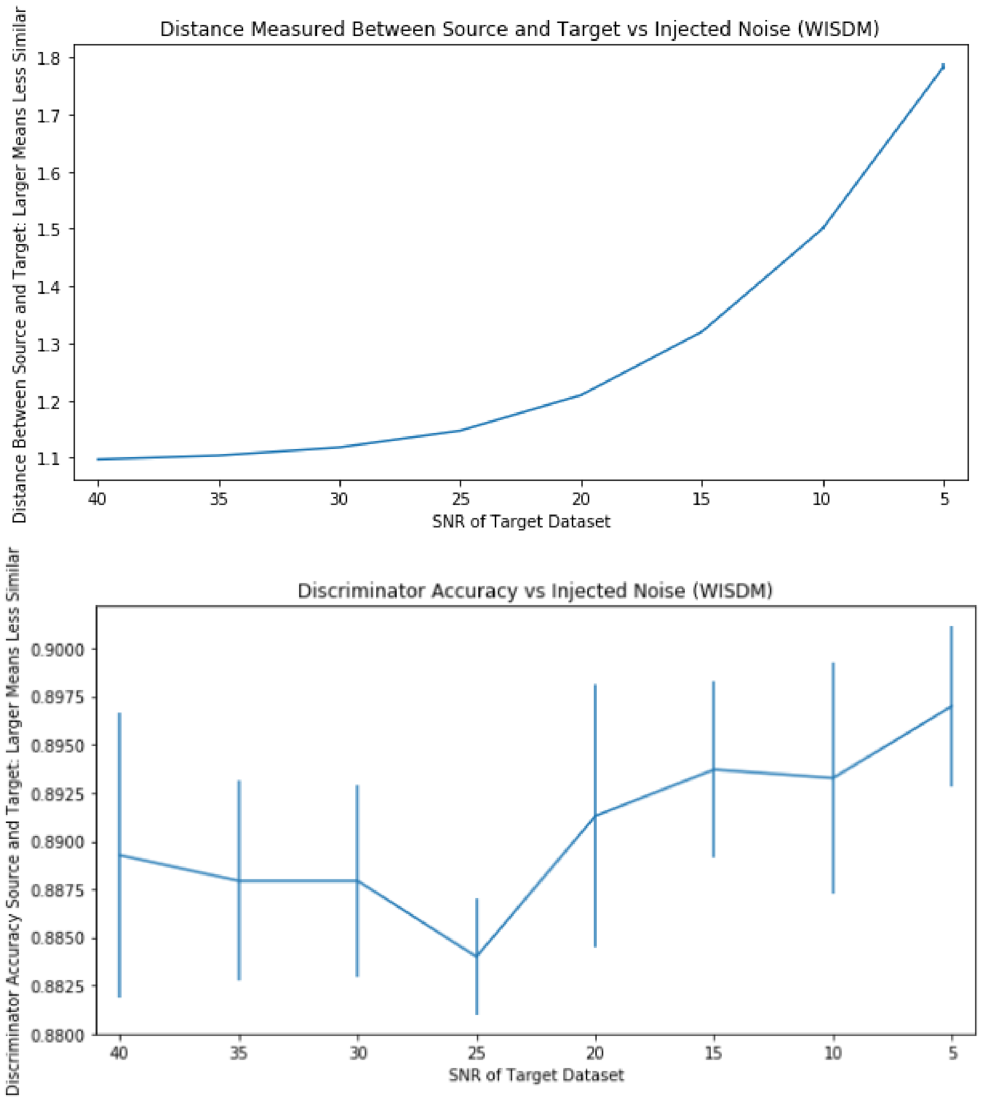


Figure 1: Comparison of the two similarity metrics to measure the similarity between the UCI HAR dataset and a noisy version of the WIDSDM dataset

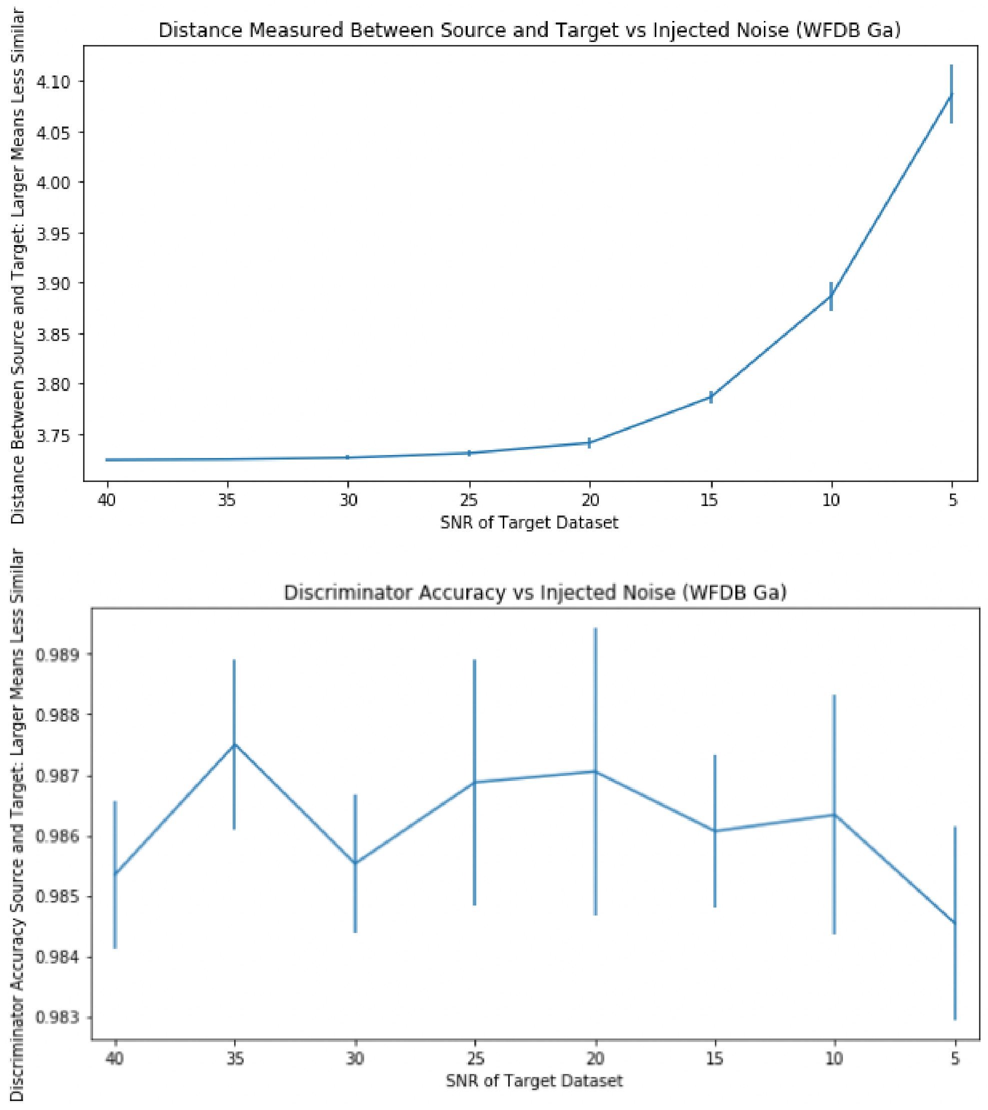


Figure 2: Comparison of the two similarity metrics to measure the similarity between the UCI HAR dataset and a noisy version of the WFDB GA dataset

Dataset	**Manhattan Distance**	**Discriminator Accuracy**	**Transfer Learning Boost**	**Area Between Curves**
DSA (Torso)	1.35 ±0.02	0.949 ±0.002	0.36 ±0.06	53 ±6
DSA (Right Arm)	0.89 ±0.01	0.905 ±0.0009	0.4 ±0.1	47 ±7
DSA (Left Arm)	1.19 ±0.02	0.933 ±0.003	0.38 ±0.06	42 ±8
DSA (Right Leg)	1.362 ±0.004	0.974 ±0.004	0.22 ±0.1	21 ±4
DSA (Left Leg)	1.925 ±0.004	0.98 ±0.003	0.41 ±0.06	42 ±8
WFDB (Ga)	3.87 ±0.07	0.987 ±0.001	-0.01 ±0.03	5 ±1
WFDB (PTB)	3.57 ±0.05	0.979 ±0.004	0.05 ±0.03	8 ±5
WISDM	1.376 ±0.006	0.893 ±0.003	0.41 ±0.09	33 ±4
WIDSM (Gyroscope)	1.521 ±0.005	0.933 ±0.005	0.38 ±0.07	26 ±4
WISDM (Watch)	1.815 ±0.002	0.963 ±0.004	0.44 ±0.06	34 ±1
WISDM (Watch and Gyroscope)	1.454 ±0.006	0.897 ±0.006	0.26 ±0.09	21 ±6

Table 1: Dataset similarity results at 800 examples

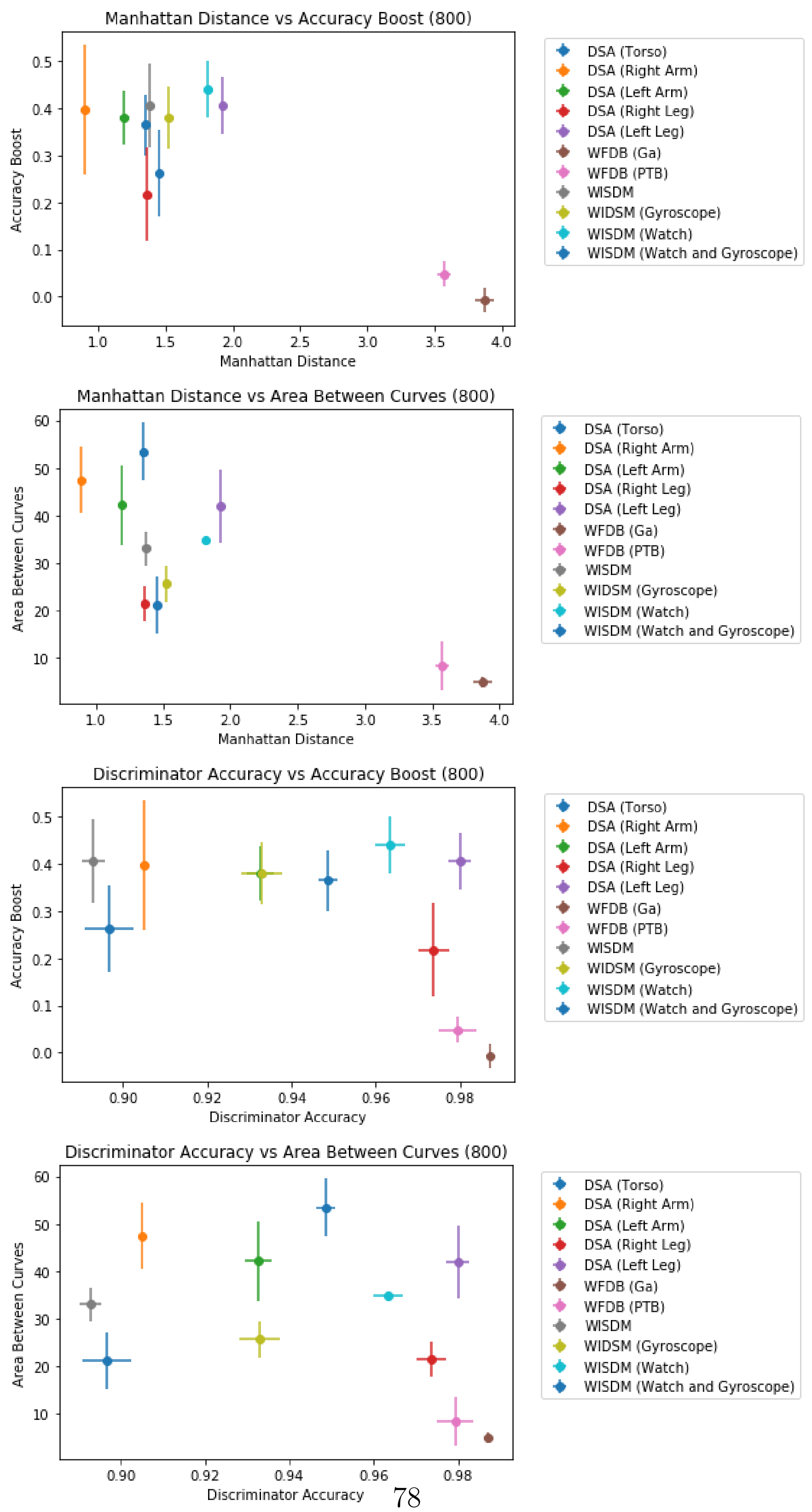


Figure 1: Dataset similarity results at 800 examples

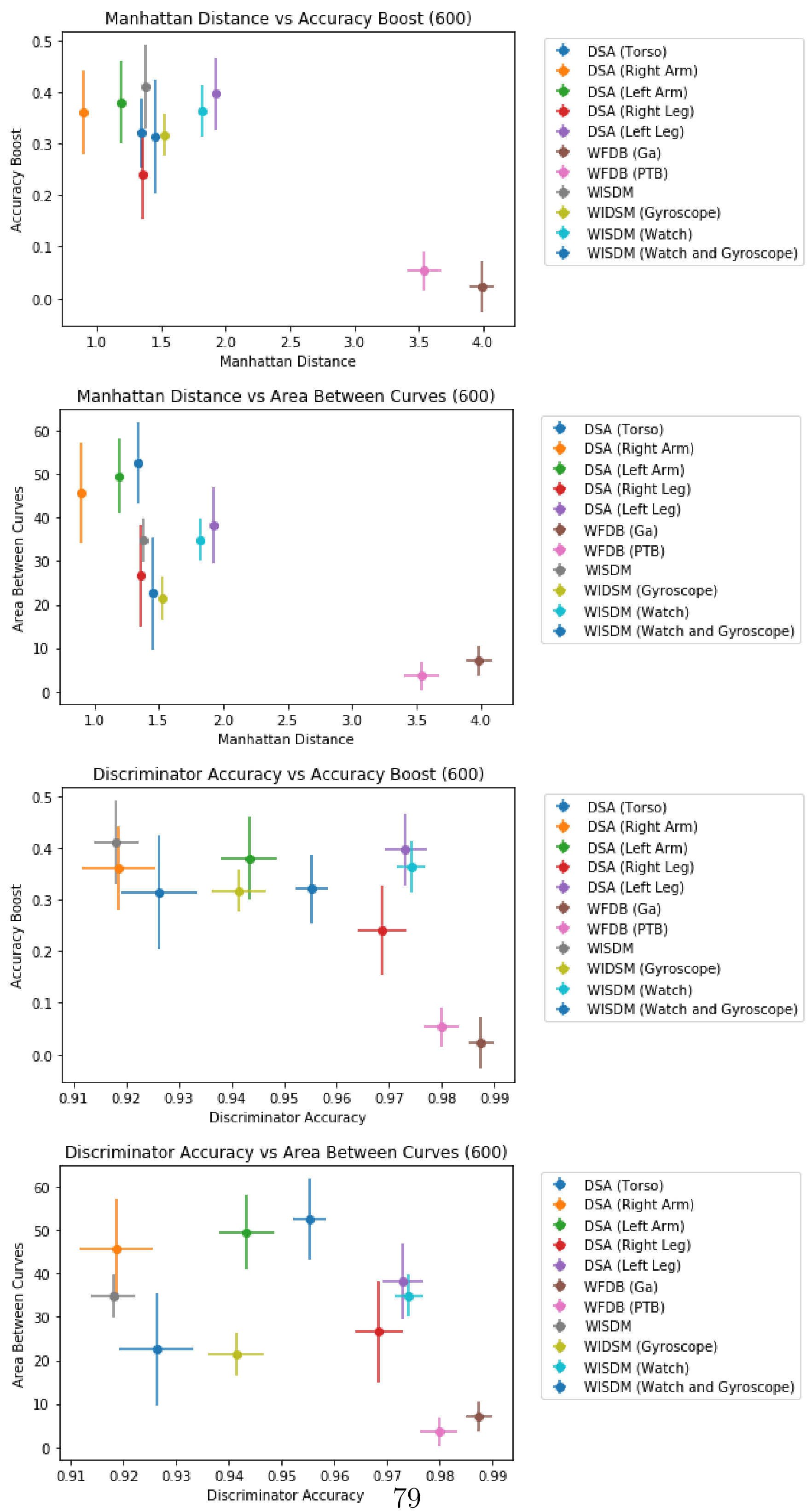


Figure 2: Dataset similarity results at 600 examples

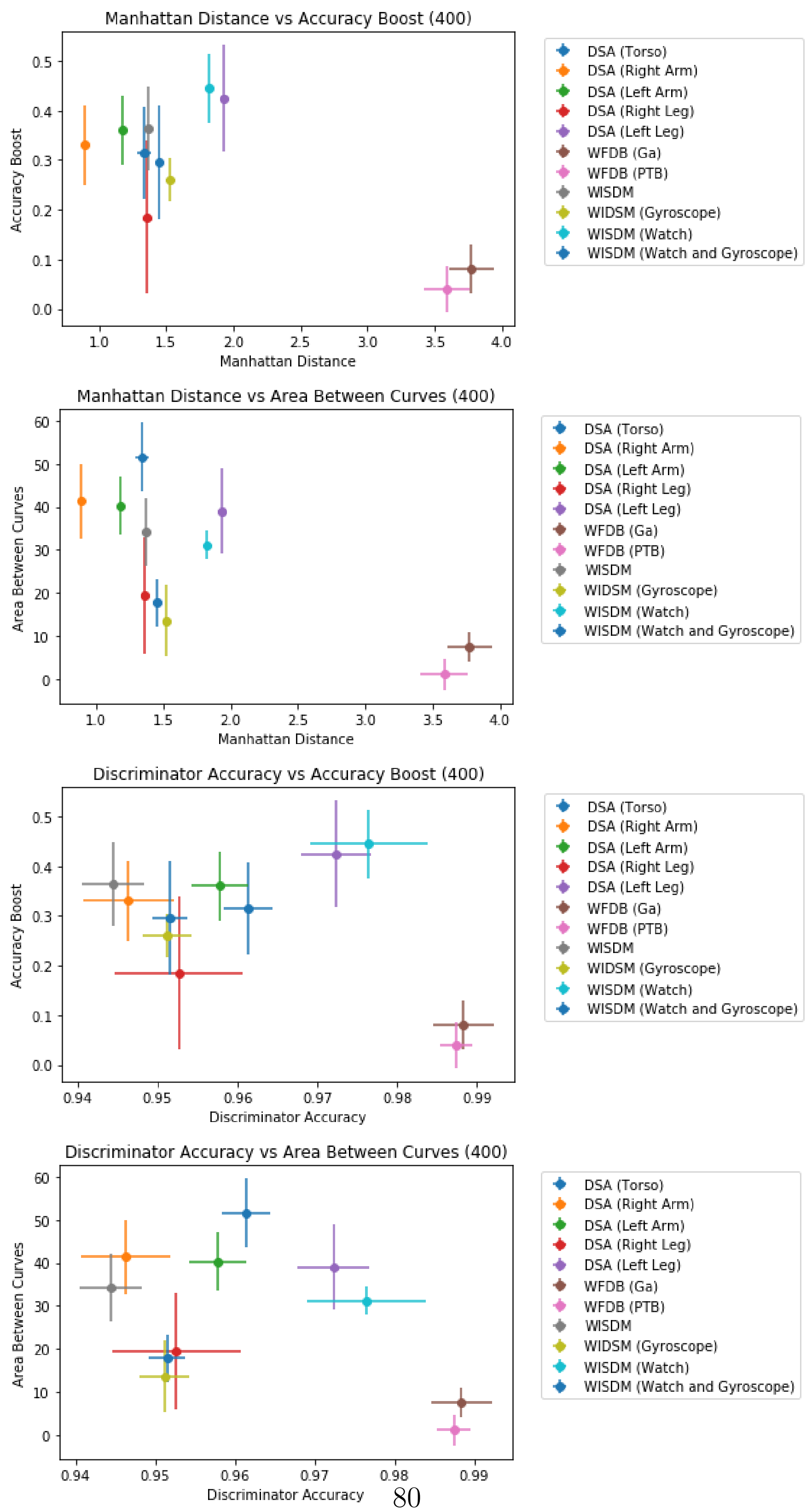


Figure 3: Dataset similarity results at 400 examples

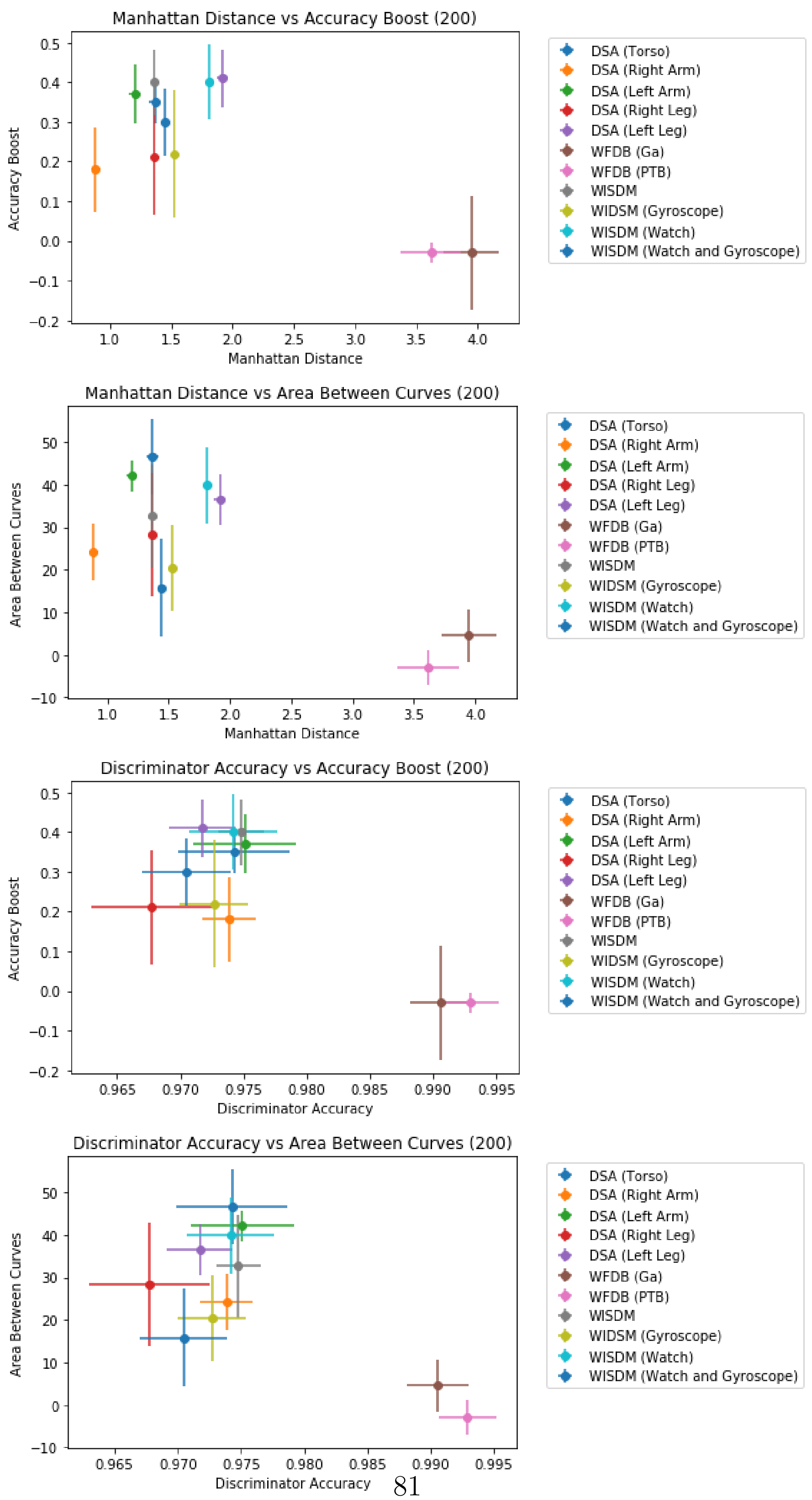


Figure 4: Dataset similarity results at 200 examples

Dataset	**Manhattan Distance**	**Discriminator Accuracy**	**Transfer Learning Boost**	**Area Between Curves**
DSA (Torso)	1.34 ±0.02	0.955 ±0.003	0.32 ±0.07	53 ±9
DSA (Right Arm)	0.897 ±0.006	0.919 ±0.007	0.36 ±0.08	46 ±10
DSA (Left Arm)	1.19 ±0.02	0.943 ±0.005	0.38 ±0.08	50 ±8
DSA (Right Leg)	1.362 ±0.008	0.969 ±0.005	0.24 ±0.09	26 ±10
DSA (Left Leg)	1.925 ±0.004	0.973 ±0.004	0.4 ±0.07	38 ±9
WFDB (Ga)	3.99 ±0.1	0.987 ±0.002	0.02 ±0.05	7 ±3
WFDB (PTB)	3.5 ±0.1	0.98 ±0.003	0.05 ±0.04	4 ±3
WISDM	1.38 ±0.003	0.918 ±0.004	0.41 ±0.08	35 ±5
WIDSM (Gyroscope)	1.524 ±0.005	0.941 ±0.005	0.32 ±0.04	21 ±5
WISDM (Watch)	1.82 ±0.01	0.974 ±0.003	0.36 ±0.05	35 ±5
WISDM (Watch and Gyroscope)	1.455 ±0.006	0.926 ±0.007	0.3 ±0.1	23 ±10

Table 2: Dataset similarity results at 600 examples

Dataset	**Manhattan Distance**	**Discriminator Accuracy**	**Transfer Learning Boost**	**Area Between Curves**
DSA (Torso)	1.34 ±0.05	0.961 ±0.003	0.32 ±0.09	52 ±8
DSA (Right Arm)	0.89 ±0.01	0.946 ±0.006	0.33 ±0.08	41 ±9
DSA (Left Arm)	1.18 ±0.03	0.958 ±0.004	0.36 ±0.07	40 ±7
DSA (Right Leg)	1.36 ±0.01	0.953 ±0.008	0.2 ±0.2	20 ±10
DSA (Left Leg)	1.93 ±0.02	0.972 ±0.004	0.4 ±0.1	40 ±10
WFDB (Ga)	3.8 ±0.2	0.988 ±0.004	0.08 ±0.05	8 ±3
WFDB (PTB)	3.6 ±0.2	0.987 ±0.002	0.04 ±0.05	1 ±4
WISDM	1.367 ±0.006	0.944 ±0.004	0.36 ±0.08	34 ±8
WIDSM (Gyroscope)	1.525 ±0.009	0.951 ±0.003	0.26 ±0.04	14 ±8
WISDM (Watch)	1.82 ±0.02	0.976 ±0.007	0.45 ±0.07	31 ±3
WISDM (Watch and Gyroscope)	1.452 ±0.005	0.951 ±0.002	0.3 ±0.1	18 ±6

Table 3: Dataset similarity results at 400 examples

Dataset	**Manhattan Distance**	**Discriminator Accuracy**	**Transfer Learning Boost**	**Area Between Curves**
DSA (Torso)	1.37 ±0.05	0.974 ±0.004	0.35 ±0.05	47 ±9
DSA (Right Arm)	0.89 ±0.04	0.974 ±0.002	0.2 ±0.1	24 ±7
DSA (Left Arm)	1.2 ±0.04	0.975 ±0.004	0.37 ±0.07	42 ±4
DSA (Right Leg)	1.364 ±0.009	0.968 ±0.005	0.2 ±0.1	30 ±10
DSA (Left Leg)	1.92 ±0.05	0.972 ±0.003	0.41 ±0.07	36 ±6
WFDB (Ga)	3.9 ±0.2	0.991 ±0.002	0.0 ±0.1	5 ±6
WFDB (PTB)	3.6 ±0.2	0.993 ±0.002	-0.03 ±0.02	-3 ±4
WISDM	1.367 ±0.006	0.975 ±0.002	0.4 ±0.08	30 ±10
WIDSM (Gyroscope)	1.525 ±0.005	0.973 ±0.003	0.2 ±0.2	20 ±10
WISDM (Watch)	1.81 ±0.02	0.974 ±0.003	0.4 ±0.09	40 ±9
WISDM (Watch and Gyroscope)	1.44 ±0.01	0.97 ±0.003	0.3 ±0.08	15 ±9

Table 4: Dataset similarity results at 200 examples

Source Dataset	Total Accuracy	Accuracy Boost	Area Between Curves	Manhatttan Distance
No Transfer	48.3%	-	-	-
UCI HAR	47.6%	-0.7%	-0.05	7.34
UCA DSA (Left Arm)	47.3%	-1.0%	-0.005	5.42

Table 1: Back2Play Results

Bibliography

[1] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. “On the Surprising Behavior of Distance Metrics in High Dimensional Spaces”. In: *Proceedings of the 8th International Conference on Database Theory.* ICDT ’01. Berlin, Heidelberg: Springer-Verlag, 2001, pp. 420–434. ISBN: 3540414568.

[2] Erick A. Perez Alday et al. “Classification of 12-lead ECGs: the PhysioNet/Computing in Cardiology Challenge 2020”. In: *Physiological Measurement* 41.12 (Dec. 2020). Publisher: IOP Publishing, p. 124003. DOI: 10.1088/1361-6579/abc960. URL: https://doi.org/10.1088/1361-6579/abc960.

[3] Hamed Alqahtani, Manolya Kavakli-Thorne, and Gulshan Kumar. “Applications of Generative Adversarial Networks (GANs): An Updated Review”. In: *Archives of Computational Methods in Engineering* 28.2 (Dec. 2019), pp. 525–552. DOI: 10.1007/s11831-019-09388-y. URL: https://doi.org/10.1007/s11831-019-09388-y.

[4] Kerem Altun, Billur Barshan, and Orkun Tunçel. “Comparative study on classifying human activities with miniature inertial and magnetic sensors”. In: *Pattern Recognit.* 43 (2010), pp. 3605–3620.

[5] Fernando Andreotti et al. “Multichannel Sleep Stage Classification and Transfer Learning using Convolutional Neural Networks”. In: *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. ISSN: 1558-4615. July 2018, pp. 171–174. DOI: `10.1109/EMBC.2018.8512214`.

[6] A. Arnold, R. Nallapati, and W. W. Cohen. “A Comparative Study of Methods for Transductive Transfer Learning”. In: *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*. ISSN: 2375-9259. Oct. 2007, pp. 77–82. DOI: `10.1109/ICDMW.2007.109`.

[7] Dor Bank, Noam Koenigstein, and Raja Giryes. *Autoencoders*. 2021. arXiv: `2003.05991 [cs.LG]`.

[8] S. Das Bhattacharjee et al. “View-Adaptive Weighted Deep Transfer Learning for Distributed Time-Series Classification”. In: *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 1. ISSN: 0730-3157. July 2019, pp. 373–381. DOI: `10.1109/COMPSAC.2019.00061`.

[9] Yiqiang Chen et al. “FedHealth: A Federated Transfer Learning Framework for Wearable Healthcare”. In: *IEEE Intelligent Systems* 35.4 (July 2020). Conference Name: IEEE Intelligent Systems, pp. 83–93. ISSN: 1941-1294. DOI: `10.1109/MIS.2020.2988604`.

[10] François Chollet et al. *Keras*. `https://keras.io`. 2015.

[11] Ryan Clark et al. “Detecting Cardiac Abnormalities with Multi-Lead ECG Signals: A Modular Network Approach”. In: *2021 Computing in Cardiology (CinC)*. Vol. 48. 2021, pp. 1–4. DOI: `10.23919/CinC53138.2021.9662677`.

[12] U. Côté-Allard et al. "Deep Learning for Electromyographic Hand Gesture Signal Classification Using Transfer Learning". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 27.4 (Apr. 2019). Conference Name: IEEE Transactions on Neural Systems and Rehabilitation Engineering, pp. 760–771. ISSN: 1558-0210. DOI: `10.1109/TNSRE.2019.2896269`.

[13] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[14] Wan-Yu Deng, Qing-Hua Zheng, and Zhong-Min Wang. "Cross-person activity recognition using reduced kernel extreme learning machine". en. In: *Neural Networks* 53 (May 2014), pp. 1–7. ISSN: 0893-6080. DOI: `10.1016/j.neunet.2014.01.008`. URL: `http://www.sciencedirect.com/science/article/pii/S0893608014000203` (visited on 10/19/2020).

[15] *Detection of GH Pituitary Tumors Based on MNF - IEEE Conference Publication*. URL: `https://ieeexplore-ieee-org.libaccess.lib.mcmaster.ca/document/8832789` (visited on 09/14/2020).

[16] Hui Ding et al. "§hdi117, goce, peters@eecs.northwestern.edu ¶xwang, eamonn@cs.ucr.edu Northwestern University University of California, Riverside Evanston, IL 60208 Riverside, CA 92517". en. In: (), p. 11.

[17] Renjie Ding et al. "Empirical Study and Improvement on Deep Transfer Learning for Human Activity Recognition". In: *Sensors (Basel, Switzerland)* 19.1 (Dec. 2018). ISSN: 1424-8220. DOI: `10.3390/s19010057`. URL: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6339185/` (visited on 11/29/2020).

[18] Jose Juan Dominguez Veiga et al. "Feature-Free Activity Classification of Inertial Sensor Data With Machine Vision Techniques: Method, Development, and Evaluation". In: *JMIR mHealth and uHealth* 5.8 (Aug. 2017). ISSN: 2291-5222. DOI: `10.2196/mhealth.7521`. URL: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5562934/` (visited on 10/25/2020).

[19] Hauke Dose et al. "An end-to-end deep learning approach to MI-EEG signal classification for BCIs". en. In: *Expert Systems with Applications* 114 (Dec. 2018), pp. 532–542. ISSN: 09574174. DOI: `10.1016/j.eswa.2018.08.031`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0957417418305359` (visited on 10/25/2020).

[20] *Effectiveness of similarity measures in classification of time series data with intrinsic and extrinsic variability.* en-US. URL: `https://ieeexplore-ieee-org.libaccess.lib.mcmaster.ca/document/6782171/` (visited on 03/20/2021).

[21] Andre Esteva et al. "A guide to deep learning in healthcare". en. In: *Nature Medicine* 25.1 (Jan. 2019). Number: 1 Publisher: Nature Publishing Group, pp. 24–29. ISSN: 1546-170X. DOI: `10.1038/s41591-018-0316-z`. URL: `http://www.nature.com/articles/s41591-018-0316-z` (visited on 10/27/2020).

[22] Andre Esteva et al. "Dermatologist-level classification of skin cancer with deep neural networks". In: *Nature* 542.7639 (Jan. 2017), pp. 115–118. DOI: `10.1038/nature21056`. URL: `https://doi.org/10.1038/nature21056`.

[23] Hassan Ismail Fawaz et al. "Transfer learning for time series classification". In: *2018 IEEE International Conference on Big Data (Big Data)* (Dec. 2018). arXiv: 1811.01533, pp. 1367–1376. DOI: `10.1109/BigData.2018.8621990`. URL: `http://arxiv.org/abs/1811.01533` (visited on 10/26/2020).

[24] Geoffrey French, Michal Mackiewicz, and Mark H. Fisher. "Self-ensembling for domain adaptation". In: *CoRR* abs/1706.05208 (2017). arXiv: 1706.05208. URL: http://arxiv.org/abs/1706.05208.

[25] Johannes Fürnkranz. In: *Machine Learning* 27.2 (1997), pp. 139–172. DOI: 10.1023/a:1007329424533. URL: https://doi.org/10.1023/a:1007329424533.

[26] Liang Ge et al. "On handling negative transfer and imbalanced distributions in multiple source transfer learning". en. In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 7.4 (2014). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10 pp. 254–271. ISSN: 1932-1872. DOI: 10.1002/sam.11217. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/sam.11217 (visited on 10/27/2020).

[27] Ary L. Goldberger et al. "PhysioBank, PhysioToolkit, and PhysioNet". In: *Circulation* 101.23 (June 2000). DOI: 10.1161/01.cir.101.23.e215. URL: https://doi.org/10.1161%2F01.cir.101.23.e215.

[28] Félix Iglesias and Wolfgang Kastner. "Analysis of Similarity Measures in Times Series Clustering for the Discovery of Building Energy Patterns". en. In: *Energies* 6.2 (Feb. 2013). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, pp. 579–597. DOI: 10.3390/en6020579. URL: https://www.mdpi.com/1996-1073/6/2/579 (visited on 03/22/2021).

[29] T. Kailath. "The Divergence and Bhattacharyya Distance Measures in Signal Selection". In: *IEEE Transactions on Communication Technology* 15.1 (1967), pp. 52–60. DOI: 10.1109/TCOM.1967.1089532.

[30] D. C. Kale et al. "An Examination of Multivariate Time Series Hashing with Applications to Health Care". In: *2014 IEEE International Conference on Data Mining*. ISSN: 2374-8486. Dec. 2014, pp. 260–269. DOI: 10.1109/ICDM.2014.153.

[31] Yohannes Kassahun et al. "Erratum to: Surgical robotics beyond enhanced dexterity instrumentation: a survey of machine learning techniques and their role in intelligent and autonomous surgical actions". In: *International Journal of Computer Assisted Radiology and Surgery* 11.5 (Jan. 2016), pp. 847–847. DOI: 10.1007/s11548-015-1340-9. URL: https://doi.org/10.1007/s11548-015-1340-9.

[32] Aakash Kaushik. "Understanding resnet50 architecture". In: *OpenGenus Foundation. Retrieved from: https://iq. opengenus. org/resnet50-architecture* (2020).

[33] Eamonn Keogh and Chotirat Ann Ratanamahatana. "Exact indexing of dynamic time warping". en. In: *Knowledge and Information Systems* 7.3 (Mar. 2005), pp. 358–386. ISSN: 0219-3116. DOI: 10.1007/s10115-004-0154-9. URL: https://doi.org/10.1007/s10115-004-0154-9 (visited on 03/23/2021).

[34] S. Kim et al. "FCSS: Fully Convolutional Self-Similarity for Dense Semantic Correspondence". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.3 (Mar. 2019). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 581–595. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2018.2803169.

[35] Serkan Kiranyaz et al. *1D Convolutional Neural Networks and Applications: A Survey*. 2019. arXiv: 1905.03554 [eess.SP].

[36] Barbara Ann Kitchenham and Stuart Charters. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. English. Tech. rep. EBSE 2007-001. Keele University and Durham University Joint Report, July 2007. URL: `https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf`.

[37] Padmini Krishnadas and Niranjana Sampathila. "Automated Detection of Malaria implemented by Deep Learning in Pytorch". In: *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. 2021, pp. 01–05. DOI: `10.1109/CONECCT52877.2021.9622608`.

[38] Jan Kukacka, Vladimir Golkov, and Daniel Cremers. "Regularization for Deep Learning: A Taxonomy". In: *CoRR* abs/1710.10686 (2017). arXiv: `1710.10686`. URL: `http://arxiv.org/abs/1710.10686`.

[39] S. Kundu and S. Ari. "MsCNN: A Deep Learning Framework for P300-Based Brain–Computer Interface Speller". In: *IEEE Transactions on Medical Robotics and Bionics* 2.1 (Feb. 2020). Conference Name: IEEE Transactions on Medical Robotics and Bionics, pp. 86–93. ISSN: 2576-3202. DOI: `10.1109/TMRB.2019.2959559`.

[40] Chen-Yu Lee et al. "Sliced Wasserstein Discrepancy for Unsupervised Domain Adaptation". In: *arXiv:1903.04064 [cs, stat]* (Mar. 2019). arXiv: 1903.04064. URL: `http://arxiv.org/abs/1903.04064` (visited on 10/27/2020).

[41] Piyapat Leeraksakiat and Wanchalerm Pora. "Occupancy Forecasting using LSTM Neural Network and Transfer Learning". In: *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*. Phuket, Thailand: IEEE, June

2020, pp. 470–473. ISBN: 978-1-72816-486-1. DOI: 10.1109/ECTI-CON49241.2020.9158103. URL: https://ieeexplore.ieee.org/document/9158103/ (visited on 09/15/2020).

[42] Vincent Liu et al. "Length of Stay Predictions". In: *Medical Care* 48.8 (Aug. 2010), pp. 739–744. DOI: 10.1097/mlr.0b013e3181e359f3. URL: https://doi.org/10.1097/mlr.0b013e3181e359f3.

[43] *Lung Sound Recognition Algorithm Based on VGGish-BiGRU*. en-US. URL: https://ieeexplore-ieee-org.libaccess.lib.mcmaster.ca/document/8850324 (visited on 09/15/2020).

[44] P. Marteau. "Time Warp Edit Distance with Stiffness Adjustment for Time Series Matching". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.2 (Feb. 2009). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 306–318. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2008.76.

[45] Bjørn Magnus Mathisen et al. "Learning similarity measures from data". In: *Progress in Artificial Intelligence* 9.2 (Oct. 2019), pp. 129–143. ISSN: 2192-6360. DOI: 10.1007/s13748-019-00201-2. URL: http://dx.doi.org/10.1007/s13748-019-00201-2.

[46] Bjørn Magnus Mathisen et al. "Learning similarity measures from data". en. In: *Progress in Artificial Intelligence* 9.2 (June 2020), pp. 129–143. ISSN: 2192-6360. DOI: 10.1007/s13748-019-00201-2. URL: https://doi.org/10.1007/s13748-019-00201-2 (visited on 03/23/2021).

[47] Grégoire Montavon et al. "Explaining nonlinear classification decisions with deep Taylor decomposition". In: *Pattern Recognition* 65 (May 2017), pp. 211–222. DOI: 10.1016/j.patcog.2016.11.008. URL: https://doi.org/10.1016%2Fj.patcog.2016.11.008.

[48] Petr Nejedly et al. "Intracerebral EEG Artifact Identification Using Convolutional Neural Networks". en. In: *Neuroinformatics* (Aug. 2018). ISSN: 1559-0089. DOI: 10.1007/s12021-018-9397-6. URL: https://doi.org/10.1007/s12021-018-9397-6 (visited on 10/25/2020).

[49] S. J. Pan and Q. Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (Oct. 2010). Conference Name: IEEE Transactions on Knowledge and Data Engineering, pp. 1345–1359. ISSN: 1558-2191. DOI: 10.1109/TKDE.2009.191.

[50] L. Rabiner, A. Rosenberg, and S. Levinson. "Considerations in dynamic time warping algorithms for discrete word recognition". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26.6 (Dec. 1978), pp. 575–582. DOI: 10.1109/tassp.1978.1163164. URL: https://doi.org/10.1109/tassp.1978.1163164.

[51] N. M. Rad and C. Furlanello. "Applying Deep Learning to Stereotypical Motor Movement Detection in Autism Spectrum Disorders". In: *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. ISSN: 2375-9259. Dec. 2016, pp. 1235–1242. DOI: 10.1109/ICDMW.2016.0178.

[52] S. Raghu et al. "EEG based multi-class seizure type classification using convolutional neural network and transfer learning". en. In: *Neural Networks* 124 (Apr. 2020), pp. 202–212. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2020.

01.017. URL: http://www.sciencedirect.com/science/article/pii/S0893608020300198 (visited on 09/29/2020).

[53] Lamyaa Sadouk, Taoufiq Gadi, and El Hassan Essoufi. “A Novel Deep Learning Approach for Recognizing Stereotypical Motor Movements within and across Subjects on the Autism Spectrum Disorder”. In: *Computational Intelligence and Neuroscience* 2018 (July 2018). ISSN: 1687-5265. DOI: 10.1155/2018/7186762. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6077579/ (visited on 10/23/2020).

[54] Siyu Shao et al. “Highly Accurate Machine Fault Diagnosis Using Deep Transfer Learning”. In: *IEEE Transactions on Industrial Informatics* 15.4 (Apr. 2019). Conference Name: IEEE Transactions on Industrial Informatics, pp. 2446–2455. ISSN: 1941-0050. DOI: 10.1109/TII.2018.2864759.

[55] Neha Sharma, Vibhor Jain, and Anju Mishra. “An Analysis Of Convolutional Neural Networks For Image Classification”. In: *Procedia Computer Science* 132 (2018), pp. 377–384. DOI: 10.1016/j.procs.2018.05.198. URL: https://doi.org/10.1016/j.procs.2018.05.198.

[56] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 1929–1958. ISSN: 1532-4435.

[57] A. Stefan, V. Athitsos, and G. Das. “The Move-Split-Merge Metric for Time Series”. In: *IEEE Transactions on Knowledge and Data Engineering* 25.6 (June 2013). Conference Name: IEEE Transactions on Knowledge and Data Engineering, pp. 1425–1438. ISSN: 1558-2191. DOI: 10.1109/TKDE.2012.88.

[58] Masashi Sugiyama. "Divergence estimation for machine learning and signal processing". In: *2013 International Winter Workshop on Brain-Computer Interface (BCI)*. 2013, pp. 12–13. DOI: 10.1109/IWW-BCI.2013.6506611.

[59] Chuanqi Tan et al. "A Survey on Deep Transfer Learning". en. In: *Artificial Neural Networks and Machine Learning – ICANN 2018*. Ed. by Věra Kůrková et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 270–279. ISBN: 978-3-030-01424-7. DOI: 10.1007/978-3-030-01424-7_27.

[60] *UCI Machine Learning Repository: Human Activity Recognition Using Smartphones Data Set*. URL: https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones (visited on 11/30/2020).

[61] *UCI Machine Learning Repository: WISDM Smartphone and Smartwatch Activity and Biometrics Dataset Data Set*. URL: https://archive.ics.uci.edu/ml/datasets/WISDM+Smartphone+and+Smartwatch+Activity+and+Biometrics+Dataset+ (visited on 11/30/2020).

[62] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[63] W. Wang et al. "Time Series Clustering Based on Dynamic Time Warping". In: *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*. ISSN: 2327-0594. Nov. 2018, pp. 487–490. DOI: 10.1109/ICSESS.2018.8663857.

[64] Weiming Wang et al. "Sensor Fusion for Myoelectric Control Based on Deep Learning With Recurrent Convolutional Neural Networks". en. In: *Artificial Organs* 42.9 (2018). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/aor.13153, E272–E282. ISSN: 1525-1594. DOI: 10.1111/aor.13153. URL: http://onlinelibrary.wiley.com/doi/abs/10.1111/aor.13153 (visited on 10/26/2020).

[65] Yunyan Wang et al. "Image Classification Based on transfer Learning of Convolutional neural network". In: *2019 Chinese Control Conference (CCC)*. 2019, pp. 7506–7510. DOI: 10.23919/ChiCC.2019.8865179.

[66] Rikiya Yamashita et al. "Convolutional neural networks: an overview and application in radiology". en. In: *Insights into Imaging* 9.4 (Aug. 2018). Number: 4 Publisher: SpringerOpen, pp. 611–629. ISSN: 1869-4101. DOI: 10.1007/s13244-018-0639-9. URL: https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9 (visited on 10/25/2020).

[67] Yao Zhao et al. "Landsat time series clustering under modified Dynamic Time Warping". In: *2016 4th International Workshop on Earth Observation and Remote Sensing Applications (EORSA)*. July 2016, pp. 62–66. DOI: 10.1109/EORSA.2016.7552767.

[68] Shuo Zhang et al. "Anomaly Detection of Periodic Multivariate Time Series under High Acquisition Frequency Scene in IoT". In: *2020 International Conference on Data Mining Workshops (ICDMW)*. 2020, pp. 543–552. DOI: 10.1109/ICDMW51313.2020.00078.

[69] Yufeng Zheng, Hongyu Wang, and Yingguang Hao. "CNN study of convolutional neural networks in classification and feature extraction applications".

In: *Big Data II: Learning, Analytics, and Applications*. Vol. 11395. International Society for Optics and Photonics, Apr. 2020, 113950K. DOI: 10.1117/12.2560372. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11395/113950K/CNN-study-of-convolutional-neural-networks-in-classification-and-feature/10.1117/12.2560372.short (visited on 10/26/2020).

[70] Yiyun Zhou et al. "Abnormal Activity Detection in Edge Computing: A Transfer Learning Approach". In: *2020 International Conference on Computing, Networking and Communications (ICNC)*. ISSN: 2325-2626. Feb. 2020, pp. 107–111. DOI: 10.1109/ICNC47757.2020.9049794.