

AMERICAN UNIVERSITY OF BEIRUT

MACHINE LEARNING FROM LIMITED
TIME-SERIES DATA

by

REEM ABDULRAHMAN MAHMOUD

A dissertation
submitted in partial fulfillment of the requirements
for the degree of Doctorate of Philosophy
to the Department of Electrical and Computer Engineering
of Maroun Semaan Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
September 2022

AMERICAN UNIVERSITY OF BEIRUT

MACHINE LEARNING FROM LIMITED TIME-SERIES DATA

by

REEM ABDULRAHMAN MAHMOUD

Approved by:



Dr. Zaher Dawy, Professor

Chairperson of Committee

Electrical and Computer Engineering



Dr. Hazem Hajj, Professor

Advisor

Electrical and Computer Engineering



Dr. Fadi Karameh, Associate Professor

Member of Committee

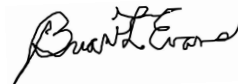
Electrical and Computer Engineering



Dr. Wassim Elhajj, Professor

Member of Committee

Computer Science



Dr. Brian Evans, Professor

Member of Committee

Electrical and Computer Engineering



Dr. Khaled Shaban, Professor

Member of Committee

Electrical and Computer Engineering



Dr. Peter Kairouz, Staff Research Scientist Member of Committee
Electrical and Computer Engineering

Date of dissertation defense: August 25, 2022

ACKNOWLEDGEMENTS

I was never a one-favorite-quote kind of person. Today, my quote of the day is “If you think you can or you think you can’t, you’re right.” as said by Henry Ford. I write this ending a four-year journey of learning, adapting to beautiful and challenging life changes, and growing as a person, a family, and a member of my community. It is my faith that got me here today. Well, my faith and an indispensable group of special individuals.

I am eternally grateful to so many who made this journey possible, but most importantly, a pleasant one.

I am grateful to incredible teachers and mentors who have helped shape me into the person I am today. In particular, I thank a particular group of mentors whose guidance was particularly invaluable during my Ph.D. journey: Dr. Hazem Hajj, the members of my committee - Dr. Zaher Dawy, Dr. Fadi Karameh, Dr. Wassim Elhajj, Dr. Brian Evans, Dr. Khaled Shaban, and Dr. Peter Kairouz - Dr. Abd-Elhamid Taha.

I am blessed to have had amazing colleagues supporting me along the way. Thank you to my AUB MIND lab buddies, Ph.D. partners, and the ECE department who shared my days with me during this journey. Thank you to my Zaka team for making the Ph.D. experience unpredictable!

I cannot do justice to my family- those with whom I share blood and those with whom I’ve come to share life: Farah and Hiba—the sisters I never knew I could have. Nadine and Haya—the people who taught me the true meaning of friendship. My brothers, uncles, aunts—the never failing support system.

To my husband, Ahmad, you are the everlasting rock I can always lean on.

To my daughter, Ayla, you are my shiniest ray of sunshine.

To my grandfather, Riad, you taught me kindness and generosity.

To my grandmother, Mariam, you taught me strength and selfishness.

To my mother, Dima, you taught me the meaning of perseverance and bravery.

To my father, Abdulrahman, you taught me unconditional love. I dedicate to you this dissertation—my life’s work.

Above it all,

الحمد لله الذي بِنِعْمَتِهِ تَتِمُّ الصَّالِحَات

ABSTRACT

OF THE DISSERTATION OF

Reem Abdulrahman Mahmoud for Doctorate of Philosophy
Major: Electrical and Computer Engineering

Title: Machine Learning from Limited Time-series Data

Time-series presents an important class of data in our everyday life and is becoming predominant with the abundance of sensors and IoT devices, which as a result has created opportunities for new machine learning (ML) applications. Unfortunately, the vast amounts of data collected are unlabeled and annotation of such data for ML becomes a challenge as it demands high monetary cost, labor, and time. This work aims at developing methods to overcome data limitations for time-series and advancing transfer learning approaches.

In the first objective of the work, we address the labeled data deficiency bottleneck when building personalized models for a group of target tasks. Most researchers have approached the problem of learning personalized models through learning a unique model per task. We present a new systematic approach for designing, evaluating, and improving Multitask Learning models, which learn multiple target tasks simultaneously and leverage information transfer across all tasks. We consider three primary design components: features capturing the time dynamics in data, similarity metrics reflecting degrees of commonality and uniqueness across entities, and generalization metrics to prevent overfitting. The framework enables the introduction of efficient new MTL models and advances the prior state-of-the-art. The approach is successfully applied and tested resulting in an MTL deep learning approach that makes use of Convolutional Neural Networks (CNN) and Gated Recurrent Units (GRU).

In the second objective of the work, we consider the case where new tasks emerge for models that had been previously trained on sufficient labeled data (referred to as ‘source’ data) but where the source data is no longer accessible, a common scenario that arises due to lack of resources or privacy and security constraints. The goal in such scenarios is to transfer knowledge from a pre-trained model to the emerging new target tasks without sacrificing performance on the source data tasks, a problem known as catastrophic forgetting. We propose a novel multi-objective

learning approach with three loss functions to minimize catastrophic forgetting, prediction error, and generalization error where label shifts exist across the source and target tasks. Under the first objective, the contributions of this dissertation are two folds. We introduce an architecture supporting the multi-objective method and targeting multitask prediction for time-series data.

In the third objective of the work, we investigate the task transferability estimation problem, which aims to provide an a priori estimate of the success of knowledge transfer between a given pre-trained model of a source task and a dataset of a new target task. Previous work has explored empirical and analytical solutions to the transferability estimation problem with remaining limitations in how the transferability relationship is defined between the source and target tasks. We present a method to show which representations extracted from a source pre-trained model are most descriptive of source and target task transferability. We build an interpretable attention network that learns the optimal combination of pre-trained model representations that hold the highest contribution to transferability across the source and target tasks. Under the second objective, the contributions of the dissertation are two folds. We introduce an attention-based transferability measure improving on state-of-the-art transferability estimation and presenting an interpretable technique to define the relationship between a source and target task.

We evaluate our proposed methods on a range of benchmark human activity recognition datasets as a sensing application as well as benchmark computer vision object detection datasets for evaluation against the state-of-the-art. Our proposed work is shown to advance state-of-the-art methods under both dissertation objectives.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	1
ABSTRACT	2
ABBREVIATIONS	10
1 Introduction	11
1.1 Research Objective 1: Multitask Learning	11
1.2 Research Objective 2: Lifelong Learning	13
1.3 Research Objective 3: Transferability Measures	15
1.4 Dissertation Contributions	16
1.5 Dissertation Outline	17
2 Background	18
2.1 Time-series Data	18
2.2 Machine Learning	19
2.3 Neural Networks	20
2.3.1 Convolutional Neural Networks	22
2.3.2 Recurrent Neural Networks	23
2.4 Transfer Learning	23
3 Literature Review	25
3.1 Traditional Transfer Learning	25
3.2 Multitask Learning	25
3.2.1 Single-task Personalized Modeling Approaches	26
3.2.2 Multitask Deep Learning Approaches	26
3.3 Catastrophic Forgetting in Neural Networks	27
3.3.1 Replay Methods	27
3.3.2 Parameter Isolation Methods	28
3.3.3 Regularization-based Methods	28
3.4 Task Transferability Estimation	29
3.4.1 Empirical Methods	29
3.4.2 Analytical Methods	30
3.5 Time-series Neural Network Architectures	30

4	Systematic Approach to Designing Multitask Learning Models for Time-series Data	31
4.1	Overview	31
4.2	Problem Definition	32
4.3	Methods	33
4.3.1	Multitask Learning Framework	33
4.3.2	Application of MTL Framework in Deep Learning for Time-series	35
4.4	Data & Experimental Setup	38
4.4.1	Datasets	38
4.4.2	Experimental Setup	40
4.5	Results & Discussion	41
4.5.1	Comparison against Baseline Models	41
4.5.2	Comparison against state-of-the-art	42
4.6	Broader Applications	43
5	Multi-objective Learning to Diminish Catastrophic Forgetting in Lifelong Transfer Learning	44
5.1	Overview	44
5.2	Problem Definition	45
5.3	Proposed Methods	45
5.3.1	Catastrophic Forgetting Minimization	47
5.3.2	Robustness with End-to-End Representation Learning	48
5.3.3	Time-series Data Architecture	50
5.4	Data & Experimental Setup	50
5.4.1	Datasets	50
5.4.2	Experimental Setup	51
5.5	Results & Discussion	52
5.5.1	Single New Task	52
5.5.2	Multiple New Tasks	55
5.5.3	Ablation Analysis	58
5.6	Broader Applications	59
6	Measures for Transferability Estimation of Pre-trained Models	60
6.1	Overview	60
6.2	Problem Definition	60
6.3	Proposed Methods	61
6.3.1	Log Expected Empirical Prediction Measure	61
6.3.2	Transferability with Multi-granular Representation Extraction	63
6.3.3	Interpretable Attention Networks for Learning Representation Importance	64
6.4	Data & Experimental Setup	66
6.4.1	Implementation Details	67
6.4.2	Datasets	67
6.4.3	Pre-trained Models	67

6.4.4	Transfer Learning Algorithms	68
6.4.5	Source-Target Settings	68
6.4.6	Baselines	68
6.5	Results & Discussion	68
6.5.1	Transferability Measures vs. Transfer Accuracy	68
6.5.2	Evaluation under the Cross-domain Setting	69
6.5.3	Evaluation against Varying Design Parameters	70
6.5.4	Interpretation of iLEEP	71
6.5.5	Applications of iLEEP and \mathcal{N} LEEP+	71
6.6	Broader Applications	72
7	Conclusion & Future Directions	73
7.1	Multitask Learning Models for Time-series Data	73
7.1.1	Summary	73
7.1.2	Open Research Directions	73
7.2	Catastrophic Forgetting in Transfer Learning	74
7.2.1	Summary	74
7.2.2	Open Research Directions	75
7.3	Measures of Transferability Estimation	75
7.3.1	Summary	75
7.3.2	Open Research Directions	75
A	Data Statistics of Chapter 5 Experiments	77
	Bibliography	79

ILLUSTRATIONS

1.1	The learning spectrum of Machine Learning approaches.	12
1.2	The traditional extremities of the transferability estimation problem.	15
1.3	The dissertation outline divided across chapters.	17
2.1	A sample time-series data sequence collected from a 3-dimensional accelerometer sensor.	19
2.2	A two-layer Neural Network architecture. [29]	21
2.3	The movement of the convolution operation in a layer of the CNN. [30]	22
2.4	A Convolutional Neural Network architecture. [30]	22
2.5	A Gated Recurrent Unit architecture. [31]	23
2.6	An example of a standard transfer learning setting of fine-tuning a pre-trained model.	24
4.1	The figure shows two user entities under two activities: walking and jumping. Both users are shown to share a common motion structure in both activities while having unique dynamics under each activity.	33
4.2	A hierarchical CNN-GRU network that is capable of handling time-series sequences effectively by capturing rich dynamic features at multiple granularity levels in the network layers.	36
4.3	A hierarchical MTL CNN-GRU network that is capable of learning unique entity models while also sharing common information across similar entities through hard-parameter.	37
4.4	The proposed MTDL-TS hierarchical convolutional-recurrent network structure is illustrated. The layers labeled Conv1, Conv2, Conv3, Conv4, dense, and GRU1 are shared across all entities, and the GRU2 and softmax layers are designed to be entity-specific, that is, learned with data that is unique to a layer’s assigned user.	38
4.5	Summary of results on the baseline pop-STD, baseline pers-STD, MTL-TS, and MTDL-TS models on ALKAN under 5 and 10 users.	41
5.1	Diagram detailing the architecture and different losses involved in the learning stage of the proposed LOMA approach.	46
5.2	Comparing LOMA performance when Λ_o is varied between [0.05,0.95] on old tasks in the objective function of Eq. 5.1	54

5.3	The plots present average performance across the 5 OPP users for which the network is fine-tuned after being pre-trained with ALKAN 20 users. The training data size of ALKAN is reduced in orders of magnitude to showcase robustness of the methods to limited pre-trained data. (a) represents fine-tuning on OPP with no label shift, and (b) represents fine-tuning on OPP with label shift.	57
6.1	An illustration of the multi-granular nature of representations that are learned across shallow to deep layers in a network. The network architecture shown is of VGG16 [34].	64
6.2	Attention network architecture for interpreting the importance of source pre-trained model representations. L is the number of layers in the source pre-trained network, P is the size of the low-dimensional input representation after being compressed through PCA, and '?' reflects an arbitrary input batch size.	65

TABLES

4.1	Comparison of OMT and MTDL-TS on OPPORTUNITY	41
4.2	Comparison of MTDL-TS on OPPORTUNITY against traditional baseline methods	42
4.3	Comparison against state-of-the-art on DSA	43
5.1	Architecture Hyperparameters	52
5.2	Experiments evaluated on 30 old tasks from UCI HAR and 4 new tasks from OPPORTUNITY. Old and new tasks have similar labels of basic locomotion activities. Results are shown in accuracy (%). . . .	53
5.3	Experiments evaluated on 30 old tasks from UCI HAR and 4 new tasks from OPPORTUNITY. Old and new tasks have different la- bels of basic locomotion activities and gesture recognition activities, respectively. Results are shown in accuracy (%).	53
5.4	Training combinations of new tasks from OPPORTUNITY simulta- neously.	55
5.5	Training all target tasks from OPPORTUNITY in sequence. The order of tasks as introduced to the model is: 1,2,3,4.	56
5.6	Training all target tasks from OPPORTUNITY in sequence. The order of tasks as introduced to the model is: 4,3,2,1.	56
5.7	Summary of ablation analysis run on 5 experimental setups. Results are shown in average accuracy (%) across all tasks.	58
6.1	Comparison of the Pearson correlation coefficients of iLEEP, \mathcal{N} LEEP+, \mathcal{N} LEEP, and LEEP on the standard setting of the source and target data having similar domain and label distributions.	69
6.2	Comparison of the Pearson correlation coefficients of iLEEP, \mathcal{N} LEEP+, \mathcal{N} LEEP, and LEEP on the cross-domain setting where the source and target data belong to different input domain distributions.	70
A.1	Summary of data statistics for ALKAN, DAS, OPPORTUNITY, and UCI HAR.	78

ABBREVIATIONS

AE	Autoencoder
CNN	Convolutional Neural Network
DL	Deep Learning
GRU	Gated Recurrent Unit
iLEEP	Interpretable LEEP
LEEP	Log Expected Empirical Prediction
\mathcal{N} LEEP	Gaussian LEEP
\mathcal{N} LEEP+	Improved Gaussian LEEP
LOMA	Lifelong Learning Multitask Autoencoder
LSTM	Long Short-Term Memory
ML	Machine Learning
MTL	Multitask Learning
NN	Neural Network
RNN	Recurrent Neural Network
TL	Transfer Learning

CHAPTER 1

INTRODUCTION

Machine learning (ML) applications have been taking the world by storm. In order to bring ML applications to reality, these ML models require vast amounts of labeled data. Time-series presents a very important class of data in our everyday life from power consumption, stock markets, and weather forecasting to context-aware sensing from IoT devices. The abundance of such time-series data is an example of the large opportunities present for new supervised ML applications. Unfortunately, most of the data collected are unlabeled and annotation of such data becomes a challenge as it demands high monetary cost, labor, and time. This has led to a limitation in the availability of labeled time-series data sources that are sufficient for training accurate ML models.

To overcome the limitations of labeled data, recent research has resulted in the successful development of pre-trained ML models to improve the performance of ML models on new tasks. Pre-trained ML models are typically fine-tuned to adapt to new tasks or information. Transfer learning presents a promising paradigm of techniques that use rich models pre-trained on large amounts of labeled data for improving performance on new target tasks that suffer from limited labeled training data. One such example is a deployed physical e-health application that has pre-trained a neural network on large amounts of labeled data from old users (or tasks). To introduce a new set of users to an existing model, labeled data for the new users must be collected. Because this process is costly in both time and effort, leveraging old knowledge from the pre-trained physical e-health neural network is important. Despite the great success of using pre-trained models to learn new tasks, there remain several challenges to overcome in order to achieve optimal transfer learning.

In this dissertation, we focus on three research challenges that address common scenarios faced in transfer learning settings:

1.1 Research Objective 1: Multitask Learning

Under the first objective of the dissertation, we address the labeled data deficiency bottleneck where prediction models for time-series data need to be developed for unique entities (e.g. group of users, stocks, or houses) to predict categories that are common across the entities (e.g. user emotions, stock exchange decisions, levels of

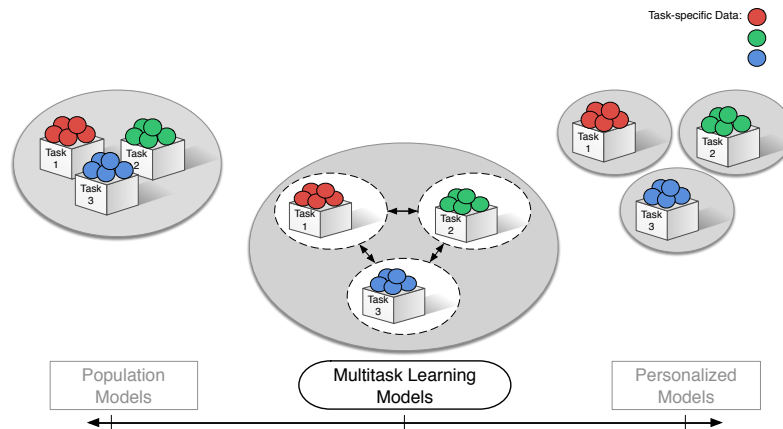


Figure 1.1: The learning spectrum of Machine Learning approaches.

power consumption). The goal is to have personalized prediction models that are unique to individual entities while achieving positive knowledge transfer by exploiting the available commonality across entities.

Personalization in modeling is motivated by the fact that it is exhibited in many natural human interactions. As an illustration, consider human activity recognition for individual users, such as walking or climbing stairs. These high-level activities include information patterns that can be understood as shared group behavior and other patterns that can be recognized as personalized to each user. On the other hand, different people exhibit unique identifiers in low-level activities such as posture, stride, and arm movements, since unique individuals vary with age, body builds, and health conditions [1]. Accurate prediction from time-series sensing data should be personalized to capture the unique descriptors of an individual or entity being learned in a model. In practice, however, the limited size of personalized labeled datasets constitutes a bottleneck in achieving the desired performance in ML models.

Most researchers have approached the problem of learning personalized models by one of two extreme approaches, population or personalized models, as shown in Fig. 1.1. *Population models* employ data collected from multiple entities to learn one unified prediction model for all entities [2], [3], while *personalized models* learn separate prediction models from each unique entity’s dataset [4], [5]. In both extreme cases, one task is learned at a time from the data. As a result, these models belong to a class of models called single task learning (STL). On the other hand, in the middle region of Fig. 1.1, multitask learning (MTL) models, introduced by R. Caruana [6], offer a tradeoff between both extremes where a compromise between specificity and generality of a model is achieved by learning personalized entity models while allowing for information sharing across entities. MTL is a type of TL that learns multiple target tasks simultaneously, leveraging their common information to improve the performance of all tasks equally. Several works have highlighted the superiority of MTL modeling in comparison to traditional STL methods, including the work of X. Sun et al. [7] and P. Lui et al. [8]. Under the MTL scenario, we

optimize a model for multiple target tasks in parallel, that is, there is no source task present and knowledge transfer is done across the target tasks in the model being optimized.

While previous work sets strong grounds for MTL models, we present a new systematic approach for designing, evaluating, and improving MTL models. This work proposes a framework for systematically investigating and building accurate personalized time-series MTL models while considering three primary design components: features capturing the time dynamics in data, similarity metrics reflecting degrees of commonality and uniqueness across entities, and generalization metrics to prevent overfitting. The framework enables the introduction of efficient new MTL models and advances the prior state-of-the-art. The approach is successfully applied and tested resulting in an MTL deep learning approach that makes use of Convolutional Neural Networks (CNN) and Gated Recurrent Units (GRU). The proposed model is tested on the classification task of human activity recognition from sensing devices and evaluated on benchmark datasets of OPPORTUNITY [9], UCI HAR [10], DSA [11], as well as ALKAN [12] of human activity recognition in-the-wild. The proposed method showed superiority in comparison to previous state-of-the-art and baseline approaches, highlighting the strength of the proposed MTL systematic design approach for personalized modeling of time-series data.

The work under this objective served as preliminary work and paved the way for open research problems that we address in the second and third objectives of the dissertation presented in Sections 1.2 and 1.3, respectively.

1.2 Research Objective 2: Lifelong Learning

Given an existing network trained on a number of pre-defined tasks, a key challenge is to devise a method that can effectively update the model to handle new tasks while leveraging and preserving old knowledge from pre-trained tasks. Traditional fine-tuning has been heavily reported in the literature, proving to be very effective in many applications. However, once a network is fine-tuned on a new (or *target*) task, it is no longer concerned with maintaining performance on old (or *source*) tasks. Federated Learning (FL) settings are an example of where preserving old source task performance is important. In FL, a large number of users are leveraging a global trained ML model on a cloud server. Moreover, these users participate in updating the global model using local model updates through continuous rounds of decentralized global model optimization. In every training round, a new set of users are selected to update the global model. The users are not identifiable, and the FL setting does not collect user data to a central location. As a result, every round that the global model is updated with new user updates, the old user model training is lost. Consequently, FL settings are an ideal scenario in practice today where overcoming catastrophic forgetting can help maintain the global model’s personalized performance for all participating users, old and new.

In a dynamic environment where new tasks keep emerging, the continuous transfer of knowledge risks forgetting previously learned tasks while fine-tuning the pre-trained model for new tasks, a phenomenon known as catastrophic forgetting. In

such an environment of continuous learning, also called Lifelong learning (LL), it is ideal to maintain performance on the previously trained source tasks while meeting the needs of new incoming target tasks. A recent survey classified methods of LL into three categories: replay, regularization-based, and parameter isolation methods [13]. Since we are focusing on time-series applications, where IoT and edge devices present low-resource challenges, we focus on regularization-based LL methods, where data storage and computational capacity requirements for such methods are suitable for low-resource settings.

In the second objective of the dissertation, we consider the case where new additional tasks emerge for models that had been previously trained on large labeled data (referred to as ‘source’ data) and where the source data is no longer accessible. Given that sensing data are abundantly collected on edge devices (e.g. smartphones, appliances, etc.) where resources are naturally limited and privacy/security concerns emerge, we assume that data from previously trained tasks is no longer available. Consequently, the goal in such a scenario is to transfer knowledge from the pre-trained model to create an updated model for the emerging new target tasks.

Catastrophic forgetting is defined as a Neural Network’s (NN) tendency to forget old information learned in its parameters when new information is introduced [14]. To overcome the problem of catastrophic forgetting, Li and Hoiem proposed a method called Learning without Forgetting (LwF) to retain memory on old tasks trained in a NN while learning new tasks through a hybrid multitask and knowledge distillation network [15]. Their approach showed state of the art performance but was sensitive to distribution shifts between the old and new tasks. Rannen, et al. [16] extended the work of Li and Hoiem [17] by introducing autoencoders that constrain new tasks’ feature space to minimize catastrophic forgetting between old and new tasks. Both prior work focused on the development of architectures for computer vision modalities which are not directly applicable to time-series data.

In this work, we address the problem of minimizing catastrophic forgetting for time-series applications and the challenge of handling output distribution shifts between old and new tasks. As an example, consider the case of a task being a user trained in our network, a label shift exists when old user output labels are locomotion activities (e.g., walking), whereas new user output labels are hand gesture activities (e.g., close fridge). We propose a multi-objective learning method with three loss functions to minimize catastrophic forgetting, prediction error, and errors in generalizing across label shifts, simultaneously. We present a hybrid approach that makes use of knowledge distillation and multitask autoencoders to improve the learning of new tasks with limited labeled data while minimizing catastrophic forgetting of old tasks trained on data that are no longer accessible. We evaluated our proposed approach for learning multiple new tasks under two settings, namely learning in parallel versus in sequence. For cases where the network needs to learn multiple new tasks sequentially, we show that the sequence in which tasks are learned matters and that starting with tasks that have the best individual performance results in optimal average performance in the network. The proposed work was evaluated on four benchmark human activity recognition datasets including ALKAN [12], DAS [18], OPPORTUNITY [19], and UCI HAR [20]. The datasets were collected from

mobile sensing devices. Our experimental results showed a reduced occurrence of catastrophic forgetting in neural network architectures compared to previous LwF state-of-the-art loss functions and traditional methods of fine-tuning.

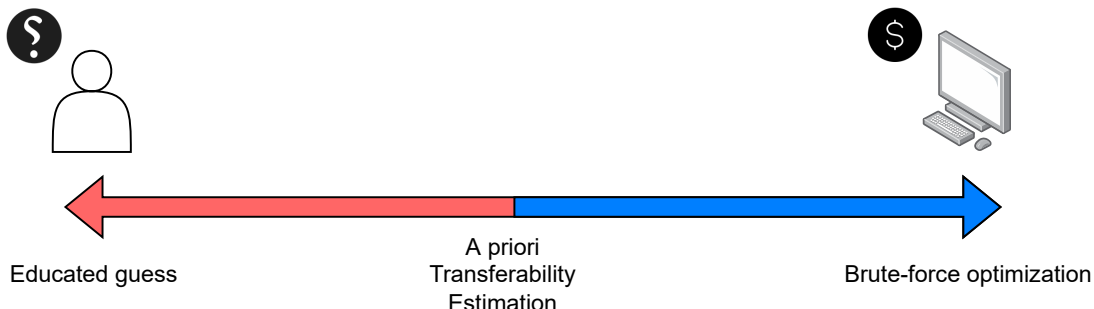


Figure 1.2: The traditional extremities of the transferability estimation problem.

1.3 Research Objective 3: Transferability Measures

In the second objective of the dissertation, we consider the case where given a pre-trained model that had been previously trained on large labeled data, we need to determine if the pre-trained model will successfully transfer knowledge to the new task and improve its performance. Ultimately, the goal in such a scenario is to determine if a source pre-trained model is able to successfully transfer knowledge to a new target task model.

Previous methods for determining if a pre-trained model is selected for transfer to a target task fall into two extreme ends as shown in Figure 1.2. On one end, an educated guess is taken to determine which source pre-trained model is most closely related to the target task. There are two unreliable assumptions made with this approach. The first assumption is that there is a clear definition of ‘relatedness’ between a source pre-trained model and target task. The second assumption is that the features extracted by a NN are interpretable and similar to the features extracted by the mind human. On the other extreme, a brute-force search over a pool of pre-trained models can help identify the best source pre-trained model by optimizing each pre-trained model for the target task and quantitatively assessing which model achieves highest transfer accuracy after being trained on the target task [21]. Although accurate, a brute-force search requires heavy optimizations that typically take place on highly complex, deep neural networks. Thus, this technique is very computationally expensive.

To help with choosing the most adequate pre-trained model, the transferability estimation problem has become more prevalent in recent literature. The goal is to measure the amount of information that can be transferred from a source pre-trained model to a target task model. Transferability estimation has been studied both empirically and analytically in literature. Empirical methods typically require expensive parameter optimization of the source pre-trained model [21]–[23]. On the

other end, analytical methods have devised measures that provide a priori information on transferability without the need for expensive optimization [17], [24], [25]. However, existing analytical methods in literature suffer from limitations as they either impose strong assumptions on the data distribution of the source and target tasks or they are not easily interpretable.

The recent work of C. Nguyen et al. [17] presents a state-of-the-art measure, called Log Expected Empirical Prediction (LEEP), that relaxes the assumptions on the data distribution between source and target tasks. Nevertheless, the proposed LEEP measure is restricted to tasks of a classification output nature. Y. Li et al. [25] improved on the LEEP measure by generalizing it to be applicable to tasks of different types including regression and unsupervised tasks. However, both LEEP and the improved \mathcal{N} LEEP measure [25] restrict their proposed measure to only study the representations of the target task from the final, deep layers of the source pre-trained model. Both prior works have developed analytical measures of transferability but have not fully explored which layers in a NN are most relevant for assessing the transferability relationship between a source pre-trained model and target task dataset.

We propose a new method to learn which NN layers in a source pre-trained model are most descriptive of source and target task transferability, thus, eliminating the restriction on relying on final model outputs only. Our proposed interpretable measure, iLEEP, uses an attention-based network to learn the optimal combination of pre-trained model representations that hold the highest contribution to transferability across the source and target tasks.

The proposed work was evaluated on two benchmark computer vision datasets of object recognition: CIFAR10 [26] and Domainnet [27]. The datasets contain camera-captured images and images pertaining to different domains including paintings, sketches, clipart, and quick drawings. Experimental results showed that our proposed iLEEP measure advances two state-of-the-art analytical measures, LEEP and \mathcal{N} LEEP, by consistently outperforming them in transferability estimation across 120 experimental runs. Furthermore, iLEEP presents novel and intuitive interpretations of the relatedness between source and target tasks.

Our experimental results show that our proposed iLEEP measure improves on prior state-of-the-art analytical measures, LEEP and \mathcal{N} LEEP.

1.4 Dissertation Contributions

The contributions of the dissertation are five folds:

1. A systematic approach for designing and evaluating MTL models for personalized modeling from time-series data that is generic and applicable to any ML model.
2. A multi-objective learning method with three loss functions for minimizing catastrophic forgetting, prediction error, and errors in generalization across label shifts, simultaneously.

3. An architecture supporting the multi-objective method and targeting multi-task prediction for time-series data. The architecture is composed of a multi-task autoencoder network with hierarchical convolutional and recurrent layers.
4. An interpretable transferability estimation method that learns an intuitive mapping of the relationship between source and target tasks.
5. A new transferability measure, iLEEP, that learns from an attention-based network the optimal combination of target representations for transferability estimation.

1.5 Dissertation Outline

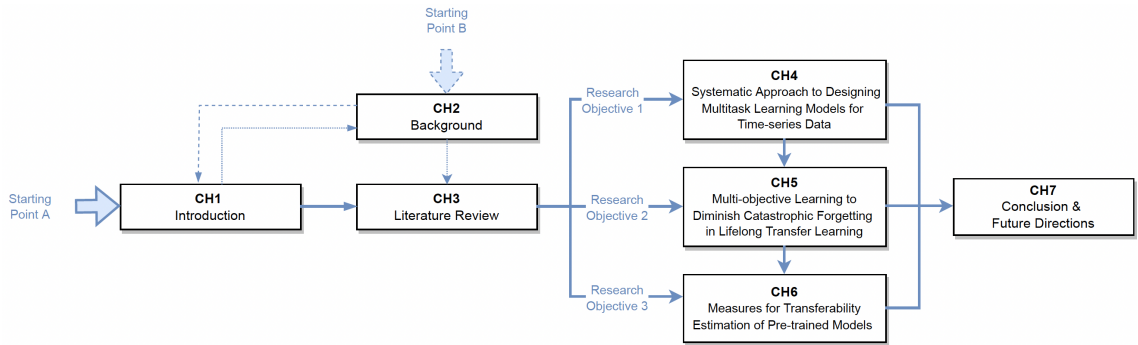


Figure 1.3: The dissertation outline divided across chapters.

The dissertation is organized as presented in Figure 1.3. In Chapter 2, we define important terms and concepts that are used throughout the dissertation. This chapter is an optional read as it only covers background information that might be common knowledge to the reader. Chapter 3 presents a review of previous work on catastrophic forgetting and the transferability estimation problem. Chapter 4 covers the proposed work of the first objective of the dissertation, which served as preliminary work and motivation for the second and third research objectives of this dissertation. Then, Chapter 5 and Chapter 6 dive into the proposed methods, experimental setup, results, and discussions under both primary objectives of the dissertation. Finally, we summarize our findings and conclude the dissertation in Chapter 7.

CHAPTER 2

BACKGROUND

This dissertation focuses on developing methods to improve the performance of supervised Machine Learning models on time-series data types with limited labeled datasets. This chapter covers an overview of the concepts and terminologies that are used throughout the dissertation.

2.1 Time-series Data

Time-series present a unique data type that exhibits unique characteristics in comparison to standard tabular data. A time-series dataset is typically composed of sequences of data collected over consecutive and equal time intervals. The time-series data is usually accompanied by a timestamp alongside the recorded samples to preserve the time element of the data. Some examples of time-series data are the weather temperature, stock market prices, and road traffic collected over a given time period. Moreover, data collected through sensing devices are classified as time-series data types. Figure 4.1 represents a time-series sequence of locomotion activities recorded from an accelerometer sensor on a human subject.

Time-series data are subject to different processing requirements due to several unique characteristics. Some of the most common characteristics of time-series are that they (1) have a **dynamic nature** in how the data evolves over time, (2) show poor ML performance under the assumption that the data is identically and independently distributed (IID), which is a typically valid assumption with other data types, and (3) come from a generative distribution that could change over time, which is referred to as the effect of *non-stationarity* in the data sequences. An IID dataset describes a setting where the collection of random variables in the dataset share a similar generative probability distribution and are assumed to be mutually independent.

In this dissertation, we consider the dynamic nature of the time-series sequences by considering ML models that can effectively learn from the dynamic evolvement of the data samples over time. Moreover, the presented time-series datasets in the experiments of this dissertation are processed to satisfy the IID and stationarity assumptions.

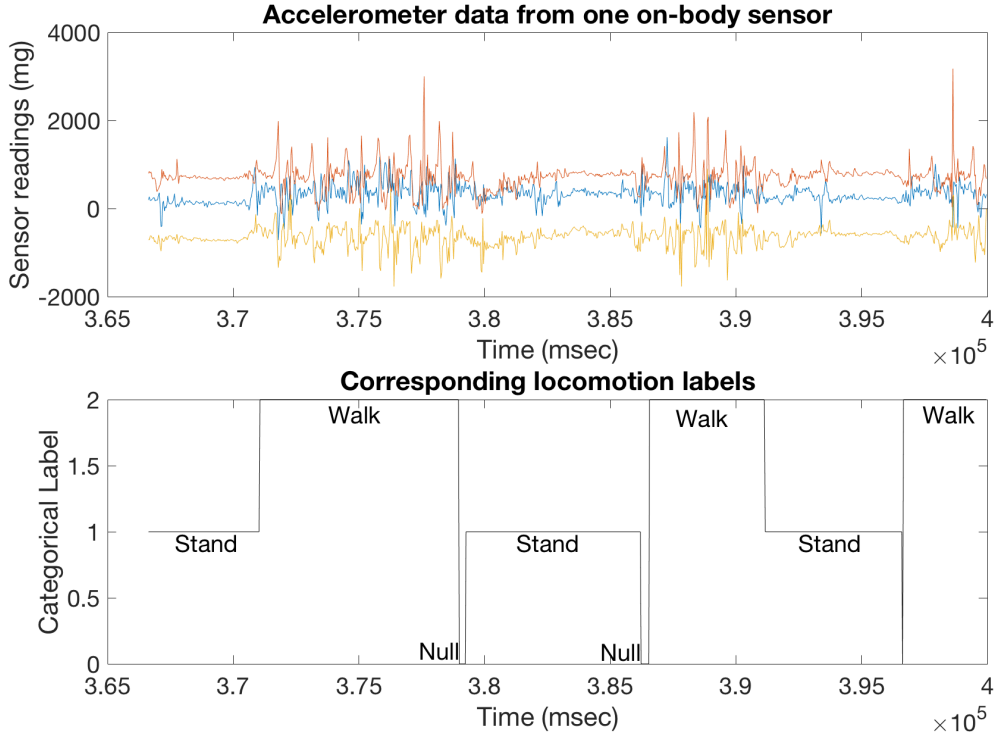


Figure 2.1: A sample time-series data sequence collected from a 3-dimensional accelerometer sensor.

2.2 Machine Learning

Machine Learning (ML) is a field that studies designing algorithms to allow machines to extract and make automated, intelligent decisions. The decisions taken by ML models are intelligent in the sense that they cannot be programmed through traditional software. On the contrary, ML extracts patterns and information present in data that help deduce the needed decisions that would typically be carried out by a human. However, the human is incapable of studying a large data size to extract patterns, at least in an acceptable period of time. In formal terms, “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .” [28]

Supervised ML is a class of ML that focuses on extracting patterns and information from annotated datasets, that is, datasets where the desired answer is known and recorded. The **annotated dataset**, \mathcal{D} , is composed of tuples of input samples, x , and true output labels, y , such that $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$. It is this annotated dataset that reflects the experience E from which a ML model learns, specifically in the supervised class of ML models.

To build the supervised ML models, the dataset is divided into two subsets, a training subset that is used to develop the model and a testing subset that is used to evaluate the model. There are a wide variety of ML models to be used that are

appropriate given certain traits of the dataset you are working with. Nevertheless, all ML models share a common structure that involves a ML hypothesis, loss, and objective function. We elaborate on each below:

1. **ML Hypothesis:** The **hypothesis function** $g(f(\mathbf{x}), \mathbf{w})$ is a potential ML mapping function that learns the relationship between a given input and a target output label through a set of parameters \mathbf{w} , where $f(\mathbf{x})$ is an input feature mapping function that extracts a unique representation of the raw input samples \mathbf{x} .
2. **ML Loss:** The **loss function** \mathcal{L} (or cost function) is a function that maps a set of values to a single real number, which represents the "cost", or intuitively the "error", in the output of the ML model. Thus, to learn the ML model, the goal becomes to optimize the loss function such that it achieves a minimum cost value.
3. **ML Objective Function:** The **objective function** defines the target you wish to optimize for in order to learn the ML model. In essence, this objective function could simply be reduced to a chosen loss function. However, the objective function may include other factors of interest to optimize for, aside from the loss function. For example, the objective function may include a regularization term \mathfrak{R} that optimizes for model generalization as opposed to solely minimizing model error in performance from the loss function. Eq. 2.1 represents an objective function that minimizes for two terms, loss and regularization, respectively.

$$g(\mathbf{w}|\mathcal{D}, f) = \underset{w}{\operatorname{argmin}} [\mathcal{L} + \mathfrak{R}(w)] \quad (2.1)$$

In this dissertation, we shed focus on applications of ML where the annotated dataset suffers from **limited data**, such that $\mathcal{D}' = \{(x_i, y_i)\}_{i=1}^I$, where $I \ll N$. The exact definition of a limited dataset remains open in the literature. For the purposes of our experiments, we present two scenarios that serve as a representation of having a limited annotated dataset.

1. The target dataset has sample count I that is at least an order of magnitude smaller than the source dataset sample count N . This is the scenario used in the experimental setup of Chapter 5.
2. The target dataset has a sample count I that satisfies the ratio of $I/K = 10$, where I is the number of training samples and K is the number of unique class labels). This is the scenario used in the experimental setup of Chapter 6.

2.3 Neural Networks

Artificial Neural Networks, or simply **Neural Networks** (NN), are a subset of ML models that are computationally designed to mimic the human neural structure.

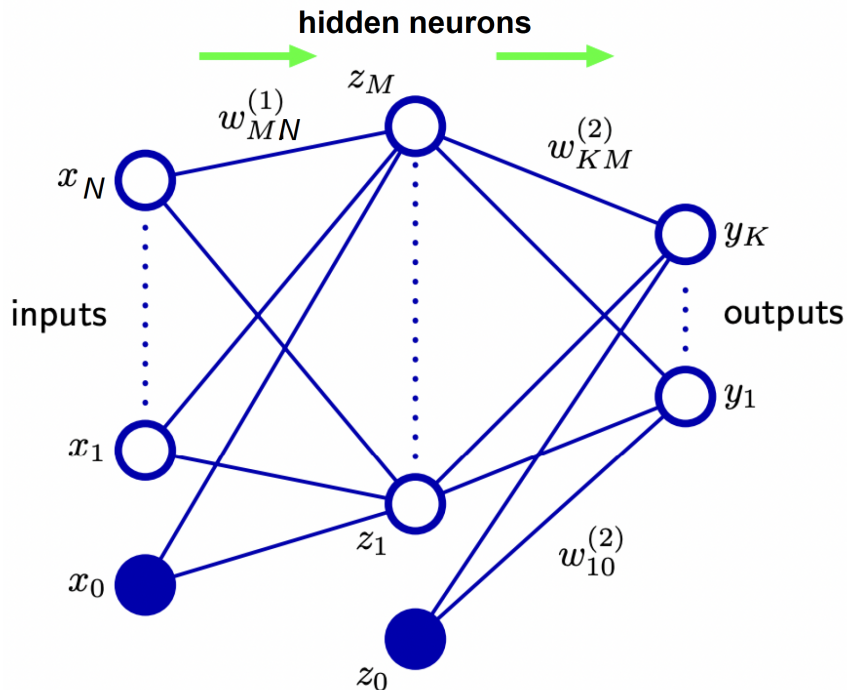


Figure 2.2: A two-layer Neural Network architecture. [29]

Consequently, a NN is composed of neurons, which are the most basic processing unit of the model. The neurons in a NN are interconnected across layers. For example, Figure 2.2 shows a NN with an input layer of N inputs, a hidden layer with M hidden units or neurons, and an output layer with K outputs, respectively from left to right. The NN is a two-layer network since the input layer does not contain weights to be trained and is usually not considered towards the layer count of a NN. The overall NN hypothesis function is shown in Eq. 2.2 [29], where σ and $h(\cdot)$ are non-linear mapping functions (or activation functions) of the output and hidden layers, respectively.

$$y_K(\mathbf{x}, \mathbf{w}) = \sigma \left(\sum_{j=1}^M w_{Kj}^{(2)} h \left(\sum_{i=1}^N w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{K0}^{(2)} \right) \quad (2.2)$$

Neural networks showed great promise in their ability to learn complex, non-linear relationships in data. Nevertheless, when it came to more complex, unstructured data types, such as images and text, new advancements were introduced to NN. These advancements made it possible for NN to extract data-specific relationships that made NN state-of-the-art models for image and text applications.

The next two sections briefly introduce Convolutional Neural Networks and Recurrent Neural Networks, which are two improved NN architectures for image and sequence data types.

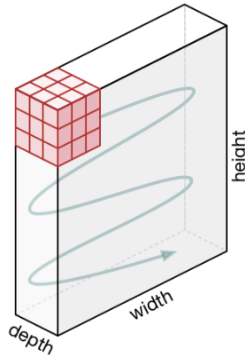


Figure 2.3: The movement of the convolution operation in a layer of the CNN. [30]

2.3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) replaced the linear combination of inputs and weights of the traditional neuron with convolution operations. These convolution operations essentially multiply a kernel, or matrix, of weights with the pixels of an input image. As a result, a CNN is a very effective ML model for processing images since it can learn relationships across neighboring pixels. Figure 2.3 shows an example of the movement of the CNN kernel across an image in a particular convolution operation.

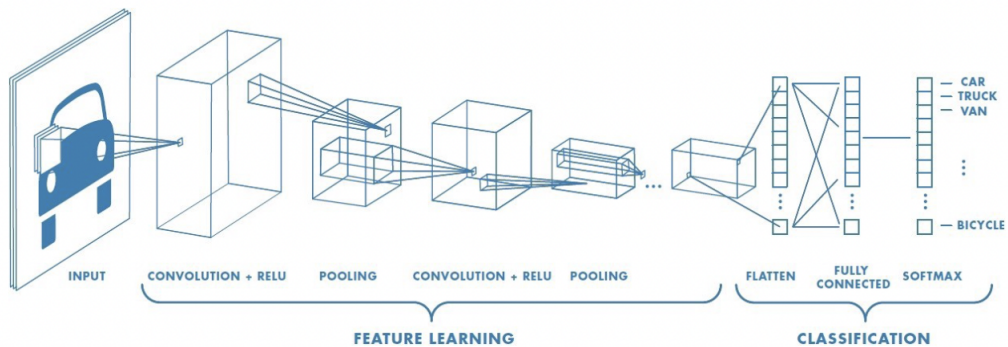


Figure 2.4: A Convolutional Neural Network architecture. [30]

Figure 2.4 presents a sample CNN architecture that is taking as input the image of a car and is being trained to label the object of the image into an output class, such as car, truck, van, or bicycle. The CNN architecture is structured similarly to a NN with convolutional layers stacked as hidden layers and composed of a number of kernels (or filters) to be convolved with the input at a given layer to extract a particular feature. The kernels are analogous to the neurons of a traditional NN layer. The convolutional layer also passes the multiplication of the layer input and kernel weights through an activation function, such as ReLU. A CNN can also include different types of layer operations, such as pooling layers to help downsample the size of the extracted features from a previous layer or fully connected layers (that

is, traditional NN layers) that are typically added at the end of a CNN to carry out the classification on the extracted features of previous convolution layers.

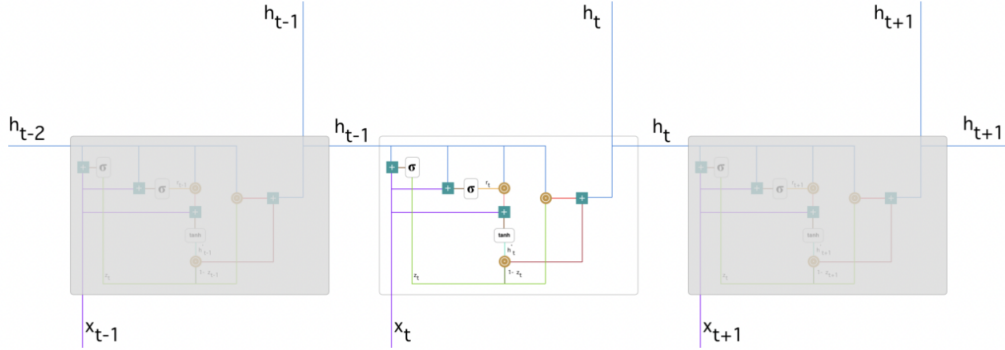


Figure 2.5: A Gated Recurrent Unit architecture. [31]

2.3.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) replaced the linear combination of inputs and weights with a cell unit that introduced computations of a given input at a given time instance, x_t , with hidden states from previous layer outputs, h_{t-1} . By involving previous layer hidden states, the RNN units are capable of maintaining an element of memory, thus making RNNs an effective NN architecture choice for sequence data types such as text and time-series. Figure 2.5 shows three RNN units over timesteps $t - 1$, t , and $t + 1$.

Gated Recurrent Units (GRUs) are RNN networks composed of GRU units that improve on basic RNN units, which were shown to suffer from exploding and vanishing gradient issues that limited the network’s ability to store long-term memory (or hidden states). GRUs were introduced to overcome the exploding and vanishing gradient problem.

2.4 Transfer Learning

Transfer Learning (TL) is a setting of ML where knowledge is leveraged from a trained ML model to learn a new ML model. Typically, the previously trained ML model, or **pre-trained model**, has been trained on a **rich source domain** and the new ML model needs to be learned for a **new target domain** that is considered to have limited data. A formal definition of TL says “Given a source domain \mathcal{D}_s and learning task T_s , a target domain \mathcal{D}_t and learning task T_t , transfer learning helps improve the learning of the target mapping function $f_t(\cdot)$ in \mathcal{D}_t using knowledge in \mathcal{D}_s and T_s , where $\mathcal{D}_s \neq \mathcal{D}_t$ and $T_s \neq T_t$.” [32] A domain \mathcal{D} consists of a feature space \mathcal{X} and a marginal probability distribution $P(X)$. Given a specific domain, a task T is defined by an output label space \mathcal{Y} and a given mapping function $f(\cdot)$. A dataset can then be composed of a single or multiple tasks typically belonging to the same domain, where each task is composed of a set of data samples $(x, f(x))$, for example,

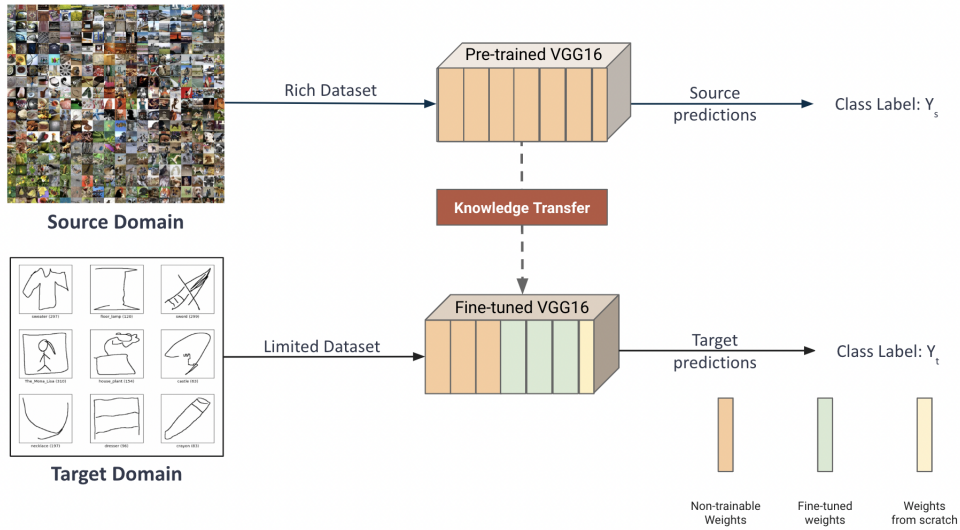


Figure 2.6: An example of a standard transfer learning setting of fine-tuning a pre-trained model.

a task could be represented by x being accelerometer recordings of human activity and $f(x)$ is the accompanying label of the locomotion activity for the represented activity of the given sample instance.

Figure 2.6 shows an example of what a transfer learning setting can look like in practice. In the top portion of the figure, a rich dataset that belongs to a source domain of camera-captured images, such as ImageNet [33], is used to train a VGG16 [34] neural network model in order to carry out predictions for the source task. Given a limited target domain, we wish to train a ML model that is of high performance. If we train the limited target domain on a new model with random weight initialization, the model is most likely going to suffer in terms of performance, since the available experience (that is, the dataset samples) for the model is limited. Thus, we can transfer knowledge from the pre-trained VGG16 model to the new target domain to improve the performance of the ML model that we wish to train.

The process of knowledge transfer is called **fine-tuning**. We can fine-tune a network by freezing all pre-trained weights in VGG16, removing the output layer from the pre-trained model, and adding and training a new output layer that has random initialization for the new target domain. We can also choose to train more layers in the model while using the pre-trained weights as starting points for the training on the new target domain, as opposed to random weight initialization. The intuition in such an example is that the new target domain is related to the rich source domain, and as a result, it can benefit from the knowledge learned by the pre-trained model. In the given example of Figure 2.6, it is valid to assume that the target domain of hand sketched images can benefit from a source domain of camera-captured images. Chapter 6 of the dissertation proposes measures that help provide apriori information about the success of a pre-trained model in transferring to a new domain.

CHAPTER 3

LITERATURE REVIEW

In this chapter, we conduct a literature survey that spans areas of traditional transfer learning where the goal is to introduce new tasks to a pre-trained neural network model [35]. We then summarize the literature on Multitask Learning (MTL) under the first objective of the work. Next, we present an overview of lifelong learning methods for overcoming the effects of catastrophic forgetting in neural networks, which is what we address in the first objective of the dissertation. We cover an overview of the literature on transferability estimation as part of the second objective of the dissertation work. Finally, we briefly review the latest trends for neural network architectures for time-series data types.

3.1 Traditional Transfer Learning

The most fundamental approach to leveraging pre-trained networks for new tasks is traditional **fine-tuning** (FT). This is where pre-trained network parameters serve as starting weight initializations when training on data for a new task. Here, a primary design choice made is which network parameters to freeze, preserving old knowledge, and which to train, learning new knowledge. In traditional FT, old task data are not accessible and no weight is given to maintaining the performance of old pre-trained tasks. On the other end of the spectrum, **multitask learning** (MTL), also known as joint learning, is another traditional approach where all old and new tasks are treated with equal importance [36]. In MTL, old and new task data are assumed to be available, and tasks are trained simultaneously with the goal of maximizing performance on all tasks [37]–[39].

In this work, old task data are assumed to be inaccessible. Thus, traditional transfer learning methods fall short in addressing our target research challenges.

3.2 Multitask Learning

This section presents three outlooks on prior work: single-task learning (STL) and Multitask Learning (MTL) modeling approaches.

3.2.1 *Single-task Personalized Modeling Approaches*

Models that fall under this category generally extract unique descriptors for users using the time-series sensing data. For example, in the work of Kwapisz et al. [4], user identification and authentication were modeled from accelerometer data of users through extracting biometric identifiers that highlighted the existence of vital unique markers in users’ sensing data. Their model was capable of identifying a user, which could then be used to customize devices to the currently active user. In another instance, Weiss et al. [40] proposed models for predicting user traits such as gender, height, and weight from sensing devices. These traits are referred to as ”soft biometrics” and form another instance that highlights the existence of unique markers in sensing data that identifies the individuality of a user.

The significance and improvement brought by building personalized models as opposed to population models, which do not account for unique entity characteristics, was demonstrated by several authors. For example, the work of Chang et al. [5] builds an activity advising system based on personalized modeling of a subject’s pace-based activities. The models are unique to every subject and provide recommendations on daily activity accounting for a subject’s personal schedule limitations. Moreover, in [41], Paradiso et al. built personalized patient models for monitoring the behavioral patterns of patients that suffer from mood disorders, such as those suffering from depressive or manic episodes. These models were built with a patient’s unique data, including sensing data sources and medical records, and the models are used to assist in alerting physicians in cases of predicted patient relapses.

In these prior approaches, the extracted features are of a static nature and do not capture dynamic characteristics in time-series data. Moreover, they made use of static modeling approaches or clustering methods to classify their targets, thus, not capturing time-dependency in prediction.

3.2.2 *Multitask Deep Learning Approaches*

There have been numerous presented work in the recent literature on MTL deep methods, and the work we present here is focused on learning sequence data. Qiu et al. [42] claim to be the first to model rainfall prediction with MTL by taking advantage of multi-site features that are fed into a CNN network with common and unique layers, that is, hard parameter sharing. Cirstea et al. [43] learn an aggregation of CNN, RNN, and autoencoder networks for MTL correlated time-series forecasting. Moreover, the recent work of Hara et al. [44] learns an MTL LSTM network through hard parameter sharing for the problem of turn-taking in conversational dialogues of human-computer interfaces.

With recent trends in Deep Learning, MTL has been incorporated into deep models through several methods. In [8], Lui et al. propose three MTL setups with recurrent neural networks (RNN), which effectively learn sequence information in time-series data. The first includes a shared network layer across tasks followed by separate task-specific layers, the second is composed of entirely separate task-specific networks that can read information from one another, and the third is a combination of both models that includes shared layers followed by task-specific

layers that can read information from one another. Their work shows superiority for the first model, validating that allowing specific layers to share information across tasks and specific layers to solely capture unique information achieves accurate performance in MTL deep networks. In [45], Harutyunyan et al. develop a model to predict multiple clinical diagnoses that are composed of a Long Short-term Memory (LSTM) network that is shared across tasks followed by a task-specific multi-layer perceptron (MLP) decoder. Yang et al. in [46] learn linguistic sequence tagging tasks together in a unified model. They form a hierarchical network composed of a shared feature encoding convolutional neural network (CNN) followed by separate task-specific CRF models, allowed to partially share parameters with each other. Their work has shown the effectiveness of using a shared CNN feature encoder followed by an MTL prediction network. In [47], Tang et al. propose an MTL model for “negatively correlated” tasks of speech recognition and speaker recognition, where tasks are not directly extracted from similar domains. Their approach is based on the idea of feeding the output of one task’s model to the input of another and has shown promising performance of deep MTL modeling with tasks that are not directly related.

M. Hassan et al. [48] built Deep Belief Networks (DBN) that are fed extracted statistical features from inertial sensors that have been passed through a Kernel PCA dimensionality reduction. In [49], S. Rokni et al. developed convolutional neural networks (CNN) for activity recognition and utilize traditional transfer learning through adjusting the weights of the network’s top layers for new tasks. A. Ignatov [50] also utilized a CNN for activity recognition; however, he investigated concatenating statistical features into the final flattened output layers of the CNN and showed improved performance with the augmented features.

While prior work sets strong ground for MTL methods, none of the prior approaches have presented a systematic approach for designing and evaluating MTL methods for personalized modeling from sensing data.

3.3 Catastrophic Forgetting in Neural Networks

Lifelong Learning (LL) solves the challenge of preserving performance on old tasks when fine-tuning a pre-trained model and can be generally classified to fall into three main categories: replay, parameter isolation, and regularization-based methods [13].

3.3.1 *Replay Methods*

Replay methods require either the storage of old task data or training a generative model that is capable of generating pseudo-samples of the old task data. In our problem, old task data was no longer accessible.

In the absence of data, replay methods focus on the generation of pseudo-samples from generative models. Robins, et al. [51] used generated output labels from old tasks to approximate task samples using constrained optimization. This approach has been shown to be limited in terms of generating pseudo-samples that can cover

the entire space of old tasks. Generative adversarial networks [52] have since been introduced and opened up the possibility of robust pseudo-sample generation [53].

Despite their success, replay methods in the absence of old task data require the maintenance of generative models for pseudo-sample generation. This is unfeasible when targeting sensing applications where models and data live on edge devices with limited storage capacity and eminent privacy concerns.

3.3.2 *Parameter Isolation Methods*

Parameter isolation methods do not require the storage of old task data. Instead, this class of methods focuses on introducing new network parameters for each introduced task so that its learned parameters are frozen during the training of new tasks, where these parameters are set as non-trainable parameters and their values are constant during training [54], [55]. This implies that the performance of old tasks is maintained during inference even after data is lost. Other works mask the network parameters of old tasks when new ones are being trained, that is, these parameters are hidden from the network during training by being set to zero [56], [57].

While parameter isolation does not require the maintenance of old tasks data, it presents computationally demanding architectures where entirely new branches are created in the network for each new task. Such computationally demanding networks would not be feasible to live on sensing edge devices due to the limited availability of resources, such as memory and battery life.

3.3.3 *Regularization-based Methods*

Similar to parameter isolation methods, regularization-based LL does not require the storage of old task data samples. However, unlike parameter isolation, catastrophic forgetting is minimized through the introduction of a regularization loss in the objective function. This implies a less computationally expensive resource requirement in compared to parameter isolation, making regularization-based LL an ideal scenario for our target problem.

One class of regularization-based method focuses on estimating task distribution from model parameters and using the distribution as prior information when learning new task data. Pfulb, et al. published one of the first studies on this method [58]. More recent works have examined how to estimate the importance of parameters for computing prior information actively during task training [59]. One major constraint in this class of regularization-based methods is the limitation in dealing with long sequences of incoming tasks, which have been shown to have increased forgetfulness of old tasks [60].

A second class of regularization-based methods focuses on data-driven methods where the foundation of these approaches is built on knowledge distillation, which is the process by which knowledge is transferred from a large model (often referred to as a teacher model) to a smaller model (often referred to as a student model) [61].

Li and Hoiem [15] proposed a Learning without Forgetting (LwF) model that constraints the output labels of the fine-tuned network to remain close to the pre-trained network recorded labels, and consequently, it does not allow the network parameters

of the old tasks to undergo large deviations from the original pre-trained network, avoiding catastrophic forgetting of old tasks. One advantage of their approach is that performance on the source domain is preserved without the need to store large amounts of data from which source tasks were trained. Their work [15] reportedly led to state of the art performance on tasks with little labeled data. Nevertheless, one limitation was its sensitivity to domain shifts between old and new tasks, where old and new tasks have different domains for classification labels. Rannen, et al. [16] extended the work in [15] by adding an autoencoder that constrains the learned feature representation of new tasks to remain close to old tasks. They were able to mitigate the sensitivity to domain shifts between old and new tasks and thus minimize catastrophic forgetting.

Prior state-of-the-art work in regularization-based LL [15], [16] has been designed and evaluated for computer vision modalities. However, the methods used for computer vision are not directly applicable to time-series applications. Moreso, state-of-the-art is shown to suffer when dealing with scenarios of data drifts between the source and target tasks. We address both challenges in our proposed work in Chapter 5.

3.4 Task Transferability Estimation

Task transferability is a crucial characteristic of transfer learning problems, where a metric is evaluated to assess to what extent source task representations can be used to enhance performance on target task training. There has been more recent work that investigates the question of task transferability in literature, and these works can be characterized into two groups: empirical methods and analytical methods.

3.4.1 *Empirical Methods*

Many prior works on transferability estimation have taken an empirical approach to studying what is being transferred during knowledge sharing [62]–[64]. These approaches would train a model on the source task(s), fine-tune the model for the target task(s), and finally carry out an assessment of task transferability.

T. Standley et al. [63] present a computational framework that exhaustively finds the best task distribution across mini networks for optimal transfer. In [65], T. Yu et al. show that poor performance in multitask learning, where several target tasks are learned simultaneously, can be attributed to learning opposing gradients during model optimization and training. They refer to this notion of learning opposing gradients as suffering from a “triad conflict”. They propose a method to eliminate the gradient interference across tasks by projecting the conflicting gradients.

The literature’s empirical methods on task transferability, though effective, require expensive optimizations and do not serve as a priori measures of transferability.

3.4.2 Analytical Methods

In contrast to the above works, there has been more recent work that focuses on measuring task transferability prior to model training, and as a result, avoiding unnecessary and expensive model optimization. A. Zamir et al. [66] present a computational structure that maps the relationship across different vision tasks into a task taxonomy space, where close task pairs are similar and thus present positive transfer of knowledge. Despite its success, the work [66] presents limitations in being specific to tasks presented in the vision domain and not applicable to broader applications.

T. Yu et al. [17] presented a scoring metric of task transferability, Log Expected Empirical Prediction (LEEP), that is computed without the need to optimize the model parameters and provides an a priori measure of transferability. LEEP is shown to achieve state-of-the-art performance on transfer learning and meta-learning tasks within the vision domain, however, the metric shows limitations when applied to cross-domain settings where the source and target input distributions are not similar. Moreover, the proposed LEEP measure is restricted to tasks of a classification output. Y. Li et al. [25] improved on the LEEP measure by generalizing it to be applicable to tasks of different types including regression and unsupervised tasks.

Nevertheless, both LEEP and the improved \mathcal{N} LEEP measure [25] restrict their proposed measure to only study the representations of the target task from the final, deep layers of the source pre-trained model. They do not investigate the relationship between feature representations at different granular levels of the network, thus, falling short in defining a clear relationship between source and target domain distributions. We address the open challenges in our proposed work in Chapter 6.

3.5 Time-series Neural Network Architectures

Neural networks have been widely adopted in the literature on time-series classification and prediction problems. Recurrent Neural Networks' (RNN) ability to handle time-dependency made them an adequate choice for modeling sequential information in time-series data [67]. More so, researchers explored convolutional (CNN) layers as well for time-series applications, where CNNs were seen to serve as adequate feature extractors of the raw time-series signals, capturing the relationship across time stamps and multiple channels of the data [68], [69]. In recent works, a common architecture used for time-series classification and prediction problems is a hybrid that combines CNN and RNN layers [38], [70]–[72].

CHAPTER 4

SYSTEMATIC APPROACH TO DESIGNING MULTITASK LEARNING MODELS FOR TIME-SERIES DATA

4.1 Overview

This chapter aims at addressing the data deficiency bottleneck where prediction models for time-series data need to be developed for unique entities (e.g. group of users, stocks, or houses) to predict categories that are common across the entities (e.g. user emotions, stock exchange decisions, levels of power consumption). The goal is to have personalized prediction models that are unique to individual entities while achieving positive knowledge transfer by exploiting the available commonality across entities. As an example, consider personalized activity recognition for individual users, where activity can be approached by considering consecutive action units to constitute high-level activities, such as walking or climbing stairs. These high-level activities include information patterns that can be understood as shared group behavior and other patterns that can be recognized as personalized to each user.

This work proposes a framework for systematically investigating and building accurate personalized time-series MTL models while considering three primary design components: features capturing the time dynamics in data, similarity metrics reflecting degrees of commonality and uniqueness across entities, and generalization metrics to prevent overfitting. The framework enables the introduction of efficient new MTL models and advancing prior state-of-the-art. The approach is successfully applied and tested resulting in two models for personalized continuous activity recognition. The first is an MTL deep learning approach that makes use of Convolutional Neural Networks (CNN) and Gated Recurrent Units (GRU), and the second advances on a previously developed MTL feature-based learning approach from the work of Sun et al. [7].

The main contribution of Chapter 4 is a systematic approach for designing and evaluating MTL models for personalized modeling from time-series data that is generic and applicable to any Machine Learning model.

The proposed systematic approach is applied under two methods, feature-based and deep learning based, which are tested on the classification task of human activity recognition from sensing devices and evaluated on four benchmark datasets including OPPORTUNITY [9], UCI HAR [10], DSA [11], as well as the in-the-wild ALKAN [12] dataset. The proposed methods show superiority in comparison to previous state-of-the-art and baseline approaches as well as competitive results when benchmarked against related work in literature, highlighting the strength of the proposed MTL systematic design approach for personalized modeling of time-series data.

The work in this chapter was published in Elsevier’s Applied Soft Computing journal under a manuscript titled ‘A Systematic Approach to Multitask Learning from Time-series Data’ [38].

4.2 Problem Definition

This work addresses the problem of designing accurate personalized multitask learning (MTL) models for prediction from time-series data. In this section, we describe the problem motivation and introduce the MTL design components that are required to formulate an accurate MTL time-series solution. We focus on the class of time-series problems of the following characteristics:

- For input, the developed model should be capable of receiving multiple time-series channels, which may include multiple sensors (e.g. accelerometer and gyroscope) and/or multi-dimensional sensors (e.g. 3-dimensional sensors with x-, y-, and z-axes). The only constraint on the input is that the different sensors and channels must be aligned in time.
- For the MTL model setup, the multiple tasks being learned simultaneously are the different entities’ personalized models. To elaborate, each entity dataset contains a set of target labels, where the labels belong to a similar category. An example of this scenario is simultaneously learning models of different users (i.e. entities) for the category of activity recognition (i.e. target labels).

Fig. 4.1 demonstrates the importance of having annotated data for different users and the possibility of knowledge transfer between them. Consider the users represented as entity t and entity t' performing two sets of activities: walking and jumping. The general motion structure of how these activities are carried out is common to different users. On the other hand, examining the still-shot of user t and user t' walking, presented in Fig. 4.1, it is possible to detect small variations in the posture of each user, which indicates descriptors that are unique to a given user’s motion. For instance, user t' is shown to have wider arm swaying motion and wider foot strides in his walking stance in comparison to user t . Another example is in the jumping difference between the two users, where user t' is shown to have higher elevation and wider arm stretch. Through recording how movements of users vary over time, it is possible to learn the unique and common dynamic structure in users’ motion patterns.

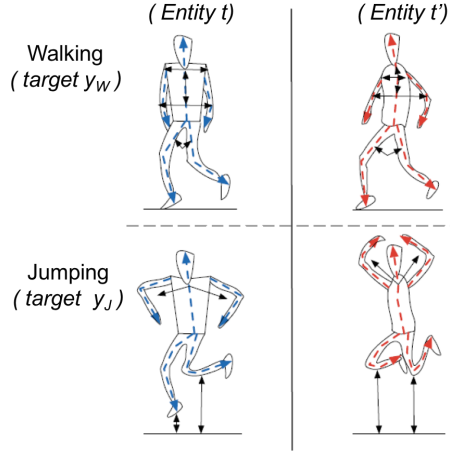


Figure 4.1: The figure shows two user entities under two activities: walking and jumping. Both users are shown to share a common motion structure in both activities while having unique dynamics under each activity.

4.3 Methods

4.3.1 Multitask Learning Framework

MTL Hypothesis

The MTL model takes as input $\mathbf{x}_{t,i}$, a data sample at index i belonging to a user t , where the input at a given instant is a vector representing time samples from an input window frame of the time-series sequence. The model then outputs $\hat{\mathbf{y}}_{t,i}$, a vector of class label predictions. Given the above, the MTL model hypothesis is defined as G_{MTL} in Eq. 4.1 for a total of T target users.

$$G_{MTL}(\mathbf{x}_{t,i}, f, W_T) = [g_1(f(\mathbf{x}_{t,i}), \mathbf{w}_1), g_2(f(\mathbf{x}_{t,i}), \mathbf{w}_2), \dots, g_T(f(\mathbf{x}_{t,i}), \mathbf{w}_T)] \quad (4.1)$$

$$W_T = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_T\}$$

G_{MTL} is defined by a given user’s input data point $\mathbf{x}_{t,i}$, a feature mapping function f , and W_T a weight matrix parameterizing the MTL model. G_{MTL} is presented as the concatenation of g_1, g_2, \dots, g_T task-dependent models, commonly referred to as *single-task learning (STL) models*, where each defines a mapping function between the model weights w_1, w_2, \dots, w_T and input features $f(\mathbf{x}_{t,i})$. Finally, the weight parameter matrix W_T is defined as being the set of all STL models’ weight parameters w_1, w_2, \dots, w_T .

MTL Loss

In MTL, we are learning a composite model G_{MTL} that is composed of a collection of personalized STL models g_1, g_2, \dots, g_T . The MTL model is taken as the interplay between the two classical approaches of personalized and population learning. Consequently, the MTL loss, presented in Eq. 4.2, is defined as the aggregate

sum of two losses: the personalized loss of user t and the impact of the loss of user t on the aggregate population loss on all other users t' .

The impact of a given user t on the losses of other users in the composite model is evaluated through introducing a similarity measure $sim_{t,t'}$ that defines the degree of relatedness between user t and user t' . As a result, $sim_{t,t'}$ defines the degree of impact user t 's loss has on other users t' .

$$\mathfrak{L}_{MTL} = \sum_{t=1}^T \left[\mathfrak{L}(t)_{STL} + \sum_{t'=1}^T (sim_{t,t'}, \mathfrak{L}(t')_{STL}) \right] \quad (4.2)$$

The STL loss is defined as the classical logistic loss for the target problem of classification from time-series windows.

$$\mathfrak{L}(t)_{STL} = -\mathbf{y}_{t,i} \log(\hat{\mathbf{y}}_{t,i}) - (1 - \mathbf{y}_{t,i}) \log(\hat{\mathbf{y}}_{t,i})$$

where recall $\hat{\mathbf{y}}_{t,i}$ is the output vector of class label predictions and $\mathbf{y}_{t,i}$ is the vector of true class labels.

MTL Objective Function

Finally, the proposed framework defines the generic objective function in Eq. 4.3 for learning an MTL personalized classification model.

$$G_{MTL}(W_T | D, f, g_1 \dots g_T, \mathfrak{L}_{MTL}) = \underset{W_T}{\operatorname{argmin}} \left[\mathfrak{L}_{MTL} + R_{MTL}(W_T) \right] \quad (4.3)$$

The MTL objective learns the weight matrix W_T given the input training dataset D , choice of feature mapping function f , STL models g_1, g_2, \dots, g_T , and MTL loss function \mathfrak{L}_{MTL} . The weight matrix W_T is learned through a minimization of the MTL loss for model accuracy and an MTL regularization for model generalizability.

In order to achieve efficient learning in the MTL solution for the target problem of personalized classification from time-series data, we propose a generic MTL framework that can apply to any ML solution through the design of three fundamental components. Given a task-dependent dataset D and the MTL solution framework of Eq. 4.3, developing the MTL solution requires the following:

1. The choice of feature mapping function f . The choice of features extracted from time-series sequences greatly impacts the learning capability of an ML model. It is crucial to design features that can capture the intrinsic dynamics in time-series sequences. This is particularly prominent in applications where accurate detection of transition events is naturally required, such as fraud detection [73].
2. The choice of MTL loss function, particularly the similarity measure $sim_{t,t'}$ that defines the learning of the MTL model. By exploiting unique and common information across tasks, the composite MTL model is bound to *learn better*, since each task is capable of benefiting from experiences (i.e. data) of other related tasks [74].

3. The choice of MTL regularizer. It is crucial to design a robustness measure $R_{MTL}(W_T)$ that is optimized aggregately across all tasks. Traditionally, STL regularizers $R_{STL}(w_t)$ would optimize the generalization of every task model separately and do not present a robust generalization metric for MTL solutions.

Finally, the choices of STL mapping functions g_1, g_2, \dots, g_T and STL loss functions $\mathcal{L}(t)_{STL}$ are dependent on the target problem. In this work, we focus on the problem of designing personalized classification models from time-series sensing data and present the applicability of the proposed MTL framework under a deep learning solution.

4.3.2 *Application of MTL Framework in Deep Learning for Time-series*

In this section, we show how the framework can be used to derive a new MTL deep learning for time-series (MTDL-TS) model composed of a hierarchical convolutional-recurrent network. We build the network components by again addressing the same three design elements of MTL models for time-series.

The input to the network can be composed of multiple sensor time-series sequences, where each sensor is labeled as S_k , $k \in \{1..K\}$. Each sensor generates readings over time and across multiple channels, thus each sensor S_k 's reading is of size $\{m^{(k)} \times n^{(k)}\}$, where $m^{(k)}$ is the number of channels and $n^{(k)}$ is the number of samples collected over time for sensor S_k . The sensor readings are fed into the network in their raw form. Each input sequence is segmented into window frames of size d . Sequence frames of all sensors $S_k, k = 1..K$ are then stacked on top of each other, with the constraint that the readings of different sensors must be aligned in time, forming a 3-dimensional input matrix to the network. The input matrix \mathbf{X} is of size $(\{B \times M \times d\})$, where B is the size of the input data batch fed into the network at a given training iteration, M is the total number of channels from all sensors S_k , and d is the size of the segmented window frames, which is set to be fixed for all sensors.

Modeling Dynamic Features

In order to effectively learn from time-series data, we need to extract general features that describe the underlying first-order statistics of the sequences as well as dynamic features that capture the time-evolving nature and statistics of the time-series sequences.

CNN's have been shown to do well on extracting relevant information from raw data in the form of feature map outputs from every convolutional layer. Shallow layers of the network capture macro-level relationships existing in the input data, while deeper convolutional layers capture more elaborate micro-level descriptors. Moreover, RNNs have been heavily used for learning sequence or temporal data and have been shown to outperform CNN's in applications with time-series data [75]. Therefore, by aggregating the CNN with an RNN network and training them as a single lumped network, the new lumped network would be capable of extracting general macro-level descriptors from the CNN and rich temporal micro-level features

from the inherent nature of the RNN cells, which are capable of storing previous states and learning long-term dependencies for new predictions.

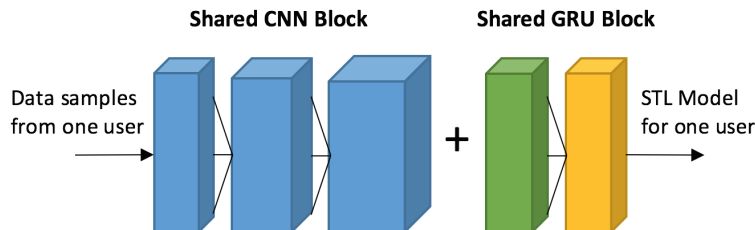


Figure 4.2: A hierarchical CNN-GRU network that is capable of handling time-series sequences effectively by capturing rich dynamic features at multiple granularity levels in the network layers.

Standard RNN networks are incapable of learning long-term dependencies within data sequences; thus, improved recurrent unit cells, such as the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), were developed to handle long-term dependencies. In this paper, we build the proposed RNN network with GRU units as they were shown to have similar performance to LSTM and in some cases even superior performance [76], [77], while being more computationally efficient with faster convergence behavior. The proposed network is composed of a multi-layer stacked GRU which are shown to outperform single-layer architectures[78].

To allow for the extraction of dynamical informative features from the input sequences, we aggregate a CNN with a GRU network. The output of the last convolutional layer is of the shape $(B \times d \times filter_{last})$, with $filter_{last}$ being the number of output channels of the last convolutional layer which is defined by the number of filters in a given layer. In order to match the shape of the output of the CNN to the accepted input shape to the upcoming RNN, we introduce a few manipulations to the CNN’s output. First, we reshape the CNN output into a 2-dimensional matrix of shape $(B.d \times filter_{last})$. We then feed the reshaped output into a dense fully-connected layer, with its output dimensions set to match the size of units present in the following RNN layer. This results in a matrix of shape $(B.d \times rnn_{size})$. The rnn_{size} variable defines the number of cell units present in the first layer of the following network. The final matrix is ready to be fed into the RNN. At this stage, a CNN network block concatenated with a GRU network block allows for effective extraction of dynamical, informative features from the data sequences. A typical architecture of the CNN-GRU network is demonstrated in Fig. 4.2, where the number of layers is arbitrarily chosen for illustrative purposes.

Modeling Similarity Between Tasks

We operate with the premise that entities share features at the macro-level but have unique characteristics at the micro-level. The macro-level common features need to be further dissected by micro-level features. This premise aligns well with MTL networks, where the first macro-network block is responsible for extracting common features, and the second micro-network block per entity is responsible for the unique features. The notion of having common layers followed by unique entity-specific layers is commonly known as "hard-parameter" sharing. An example of an

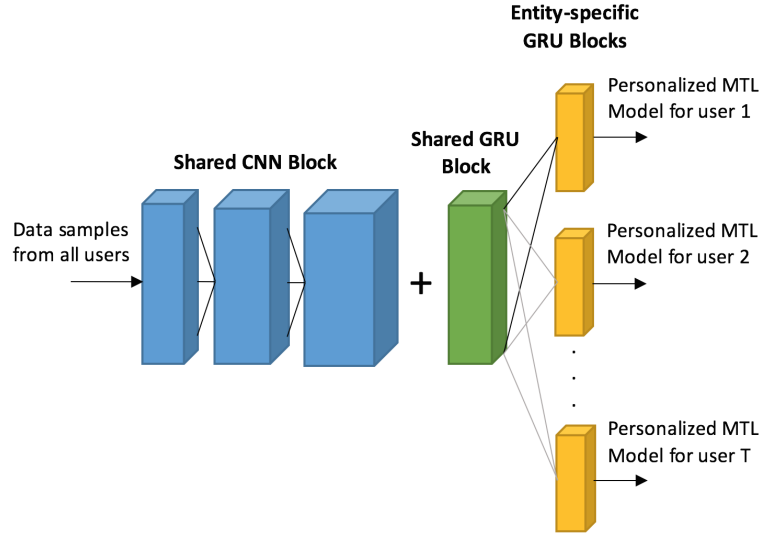


Figure 4.3: A hierarchical MTL CNN-GRU network that is capable of learning unique entity models while also sharing common information across similar entities through hard-parameter.

MTL CNN-GRU architecture is demonstrated in Fig. 4.3, where the number of layers is arbitrarily chosen for illustrative purposes. The result is an MTL model that is unique for each entity while leveraging common information across similar entities for improved personalized models.

The similarity across entities is thus captured by having the macro-block learn from all the entities’ data instances. The uniqueness, or dissimilarity, across entities is captured by having the micro network individually learn from the entity-specific data sequences only. In our proposed network, the macro-block is the CNN network aggregated with the first layer of the RNN, while the micro-blocks per entity are the entity-specific second layer RNN with the aggregated fully-connected output layer.

In the network design, each GRU layer is composed of 128 unit cells. The input to the first layer GRU is the manipulated output of the CNN layer. The output of the shared layer is then fed into the second layer GRU, which is entity-specific. The output of the final entity-specific GRU layer is fed into an augmented entity-specific dense layer that maps the output of the 128 GRU cells to the corresponding number of class labels for a given user t . The output dense layer is a softmax layer with cross-entropy minimization, which computes the final prediction probabilities of \hat{y}_t . During a given training batch iteration, only one entity-specific GRU layer is active and its weights are updated uniquely with the training data batch of that entity (or user). The next training batch iteration would hold data from another user, in which case the next entity-specific GRU layer is active and updated, and so on. This training procedure is referred to as *alternate training*.

Modeling Robustness

To model robustness in a deep network, we need to impose a learning factor that ensures model generalizability to unseen data. A common approach to improving generalization of deep networks is applying the dropout phenomenon. We apply

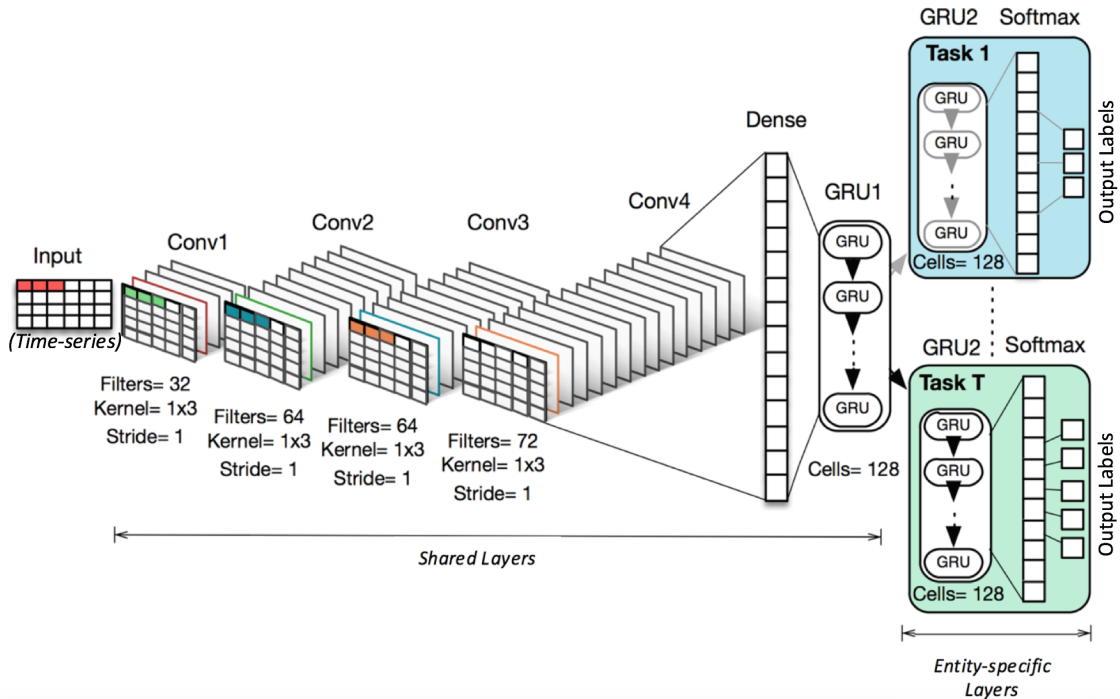


Figure 4.4: The proposed MTDL-TS hierarchical convolutional-recurrent network structure is illustrated. The layers labeled Conv1, Conv2, Conv3, Conv4, dense, and GRU1 are shared across all entities, and the GRU2 and softmax layers are designed to be entity-specific, that is, learned with data that is unique to a layer’s assigned user.

dropout to the GRU layers for enhanced generalization capability.

4.4 Data & Experimental Setup

4.4.1 Datasets

ALKAN: Human Activity Recognition in the Wild

ALKAN [12] is used as a large-scale activity dataset that was collected in a natural, realistic environment where users collect their own data using their mobile devices and throughout their regular daily lives. Data is collected using a single accelerometer sensor embedded in users’ mobile devices from which data is sampled at a 20 Hz frequency. The activity labels recorded include locomotion activities such as walking/running/jumping as well as riding a bus/train/car to accommodate transportation activities. For our experiments, we segment the sequences into window frames of 128 samples with a 50% overlap, covering frame periods of 6.4 seconds that allows enough time to resemble a user’s activity state. For a fair comparison, we follow the previously used approach for data organization of continuous activity sequences [7], which is represented by concatenating the independent sequences of a user. We use three subsets of the ALKAN dataset: 5 users, 10 users, and 20 users, where each user’s data exhibits different instance count and different label sets.

The dataset was randomly split into an 80% training set, 10% validation set and

10% testing set. The validation set was used for hyperparameter tuning through k-fold cross validation (k=10). After hyperparameters are selected, we retrain the final model on the training and validation set (total of 90% of the original dataset) and present performance results on the 10% testing set.

OPPORTUNITY: Human Activity Recognition in a Semi-controlled Environment

OPPORTUNITY [9] is a widely used benchmark activity recognition dataset that was collected in a controlled experimental setup using numerous on-body sensors and a sensor-rich environment. The time-series data is collected from the sensor modalities at a 30 Hz frequency. OPPORTUNITY provides a realistic representation of continuous activity sequences with true transitional periods captured as a user transitions from one activity to the next. The activity labels include locomotion activities (walking, running, sitting, and lying down), gesture recognition (e.g. close fridge), as well as high-level activities (e.g. making coffee). For our experiments and similar to the ALKAN data segmentation approach, we focus on locomotion activities and segment the sequences into window frames of 128 samples with a 50% overlap, covering frame periods of 4.3 seconds that allows enough time to resemble a user’s activity state. We use a subset of the data composed of 4 users and utilizes the on-body accelerometer sensor modalities to predict locomotion activity labels. We choose to use a subset with only on-body accelerometer sensors to mimic a framework that is closer to how a user would collect data throughout his daily life.

The training and testing subsets are predefined in OPPORTUNITY as it is widely used for benchmarking. We extract a validation set constituting 10% of training data for hyperparameter tuning through k-fold cross validation (k=10). After hyperparameters are selected, we retrain the final model on the predefined training set and present performance results on the predefined testing set.

UCI Human Activity Recognition (HAR)

UCI HAR [10] is a commonly used benchmark dataset on human activity recognition. The dataset is collected from 30 subjects performing six basic locomotion activities including walking, walking upstairs, walking downstairs, sitting, standing, and laying down. Data samples are collected from one smartphone allocated on the subjects’ waists, corresponding to sensing data from accelerometer and gyroscope sensors.

For this work, training and testing subsets are user-dependent. As a result, we redistribute the train and test data across each subject’s collected data samples and divide each subject’s data into 80% for model training and validation and 20% for model testing.

Daily and Sports Activities (DSA)

Another commonly used benchmark human activity recognition dataset in literature is the Daily and Sports Activities (DSA) dataset [11]. It is composed of a total of 19 activities carried out by a total of 8 subjects. The activities carried out are of basic locomotion (e.g. sitting, jumping, cycling, etc) as well as more elaborate activities (e.g. exercising on a stepper, exercising on a cross-trainer, etc.). Data is collected from 5 units of sensing devices allocated on the subject’s torso, both right and left arm, and both right and left leg. Each unit collects sensing data from a 3D

accelerometer, gyroscope, and magnetometer.

Similar to the setup for the UCI HAR dataset, training and testing subsets are user-dependent and divided as follows: each subject’s data samples are randomly divided into 80% for model training and validation and 20% for model testing.

4.4.2 *Experimental Setup*

In this section, we describe different model setups to be evaluated and discussed. We present the MTFL-TS model, MTDL-TS model, and baseline models used for benchmarking purposes.

Moreover, we use accuracy, or the recognition rate, as the primary evaluation metric of the models on the human activity recognition tasks, where the goal is to classify the correct activity label per window frame of the input time-series sequence. The *accuracy* is defined as the number of correctly classified window frames divided by the total number of frames being evaluated. To compute the *overall accuracy* of the MTL model, we average the accuracy performance across all users being learned in parallel.

MTDL-TS Setup

The proposed MTDL-TS model is implemented in Python using Google’s Tensorflow platform¹. The MTDL-TS model makes use of Tensorflow’s built-in 1-dimensional convolution layers and GRU cells. We incorporate 80% dropout across the GRU layers. We train the model through an ‘alternate training’ method, where the cost function is minimized by iteratively passing data batches from each user. At each iteration, we update the parameters of the shared layers and the layers unique to the given user, before training another user. We minimize the softmax cross-entropy cost function on the output layer of the network, where minimization is done through an Adam optimizer [79] that internally performs step size annealing and achieves convergence with time-varying objective functions. Following standard convention, the training batch size is chosen to be 128, and the model is trained for 10 epochs, where each epoch passes through the data of all users. We use the *overall accuracy* as a performance metric, where we average the accuracy performance across all users being learned in parallel in the MTL model.

Setup of Baseline Models

For MTDL-TS, the baseline models are composed of an STL convolutional-recurrent network that is similar in architecture to Fig. 4.4 while having all network layers shared across all entities. For the personalized STL deep learning (pers-STDL) model, an independent network is trained for every user’s unique dataset. Finally, for the population STL deep learning (pop-STDL) model, a single network is trained with all users’ dataset. The performance of the personalized baseline models is taken as the average accuracy of all the personalized models, and the performance of the population baseline models is taken as the accuracy of the single model itself.

¹<https://www.tensorflow.org/>

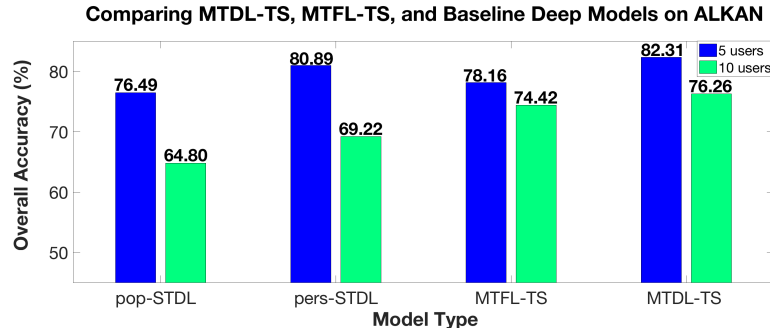


Figure 4.5: Summary of results on the baseline pop-STDL, baseline pers-STDL, MTFL-TS, and MTDL-TS models on ALKAN under 5 and 10 users.

Table 4.1: Comparison of OMT and MTDL-TS on OPPORTUNITY

Method	Overall Accuracy (%)
OMT	79.67
MTDL-TS	89.87

4.5 Results & Discussion

We evaluate our proposed MTDL-TS model under the ALKAN and OPPORTUNITY datasets. We compare MTDL-TS to baseline personalized and population STL deep learning models, pers-STDL and pop-STDL, respectively. We also compare our proposed MTDL-TS and MTFL-TS approaches in terms of accuracy and provide insight into some of the practical strengths and limitations associated with each of the two classes of machine learning models, that is, the feature-based and deep learning models.

4.5.1 Comparison against Baseline Models

Fig. 4.5 shows the performance of the proposed MTDL-TS model in comparison to baseline pers-STDL and pop-STDL baselines on two subsets of ALKAN, including 5 users and 10 users. The MTDL-TS model outperforms pers-STDL with an increase of +1.42% on the subset of 5 users and of +3.34% as the group size is increased to 10 users. As the group size increases, more benefit is brought about by MTL, taking advantage of richer shared information across the users. The MTDL-TS model also outperforms the pop-STDL baseline model by +5.82% for 5 users and +11.46% for 10 users. Once again, the impact of group size is seen on the MTL performance in comparison to baseline population models.

This proportional increase in group size and the performance of the MTL model is consistent with the results shown for the MTFL-TS model in comparison to its baselines. Another interesting result that is consistent with our expectations is that personalized baseline models have outperformed the population baseline models.

Table 4.2: Comparison of MTDL-TS on OPPORTUNITY against traditional baseline methods

Method	Overall Accuracy
MTDL-TS	0.898
LDA	0.60
QDA	0.64
NCC	0.54
1-NN	0.82
3-NN	0.83

This stresses the importance of learning unique descriptors of entities within a model. To build on that, the results on MTL validate the increased benefit of learning both unique and common information present across groups of entities.

Finally, Table 4.2 presents the performance of the proposed MTDL-TS model against traditional machine learning methods [80]. These methods include Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Nearest Centroid Classifier (NCC), and k-Nearest Neighbor for k=1 (1-NN) and k=3 (3-NN). Table 4.2 further supports the dominant results of the MTDL-TS model.

4.5.2 *Comparison against state-of-the-art*

In this section, we present additional comparative results of the MTDL-TS model of this work on two benchmark datasets: UCI HAR and DSA. Evaluation of MTDL-TS on these datasets allows for comparison against recent state-of-the-art work on human activity recognition [81], [82].

Table 4.3 shows that the proposed MTDL-TS model presents superior performance to the work of Saeedi R. et al. [81] on the DSA dataset, where the work [81] introduces a Manifold Learning-based Transfer Learning (MLTL) method. These results potentially signify the strength of an MTL solution framework where tasks are collectively optimized in parallel, taking advantage of common shared information across different tasks. Moreover, Table 4.3 shows improved performance of our MTDL-TS method in comparison to a mix of a Disc ensemble pruning and minimal redundancy and maximal relevance (mRMR) pruning algorithm (Mix), as proposed by J. Cao et al. [82].

These results highlight the significant impact of the presented systematic framework for designing personalized MTL models for sensing applications, particularly highlighting performance on the task of human activity recognition from time-series data collected from a range of sensing devices (including wearable and smartphone devices).

In summary, this work targets addressing the limitation of annotated data for personalized models by making use of multitask learning. We propose an MTL framework for accurate personalized predictions from sensing data that leverages similarity across entities being learned together for enriching limited annotated datasets and improved model generalization. The contributions of this work are

Table 4.3: Comparison against state-of-the-art on DSA

Method	Dataset	Overall Accuracy (%)
MLTL [81]	DSA	85 \pm 2
MTDL-TS	DSA	90.21
Mix [82]	OPP	88.82
MTDL-TS	OPP	89.87
MTDL-TS	UCI HAR	95.63

three folds. First, we propose a systematic method for formulating solutions of MTL time-series models composed of three primary components: dynamic features, similarity measures that capture commonality and uniqueness across entities, and a robustness metric that enhances generalization capability. Second, we propose a deep learning (MTDL-TS) solution for personalized classification from time-series under the proposed framework. Third, we show superior performance against state-of-the-art methods on three benchmark human activity datasets and one large-scale human activity dataset in-the-wild.

The MTDL-TS model showed superior performance with an improvement of up to 13% compared to the prior state-of-the-art and 11% compared to baseline STL deep networks. The presented framework was tested on personalized human activity recognition but can be generalized to a variety of other applications, such as personalized smart metering and medical diagnostics. Moreover, the presented deep learning approach can be readily applied to appropriately sampled time-series raw data.

4.6 Broader Applications

The presented methods of this chapter can be directly extended to different types of time-series problems, such as time-series forecasting applications, by using the appropriate choice of STL loss function in Eq. 4.2. As an illustration of the applicability of the proposed design framework, we present the application to a deep learning MTL model in Section 4.3.2, which can be followed similarly for any new time-series application. As a result, the systematic approach to designing the MTL models can be catered to any type of ML model and target time-series dataset by using the appropriate choice for each design element presented in Section 4.3.1.

CHAPTER 5

MULTI-OBJECTIVE LEARNING TO DIMINISH CATASTROPHIC FORGETTING IN LIFELONG TRANSFER LEARNING

5.1 Overview

In this chapter, we address the problem of overcoming catastrophic forgetting in a neural network model that has been pre-trained on a source task and is to be fine-tuned for a single or sequence of target task(s). The source task data is no longer accessible, and the goal is to fine-tune the source pre-trained model on the target task(s) while minimizing the forgetting of the source task originally trained.

The contributions of Chapter 5 are as follows:

1. A multi-objective learning method with three loss functions for minimizing catastrophic forgetting, prediction error, and errors in generalization across label shifts, simultaneously.
2. An architecture supporting the multi-objective method and targeting multi-task prediction for time-series data. The architecture is composed of a multi-task autoencoder network with hierarchical convolutional and recurrent layers.

We next present the proposed methods and evaluate the methods on four benchmark human activity recognition datasets: ALKAN [12], DAS [18], OPPORTUNITY [19], and UCI HAR [20], where data were collected from mobile sensing devices. Our experimental results showed a reduced occurrence of forgetting compared to previous state-of-the-art and traditional methods of fine-tuning.

The work in this chapter was published in ACM's Transactions on Knowledge Discovery from Data journal under a manuscript titled 'Multi-objective Learning to Overcome Catastrophic Forgetting in Time-series Applications' [83].

5.2 Problem Definition

In a transfer learning setting, we are typically interested in using a source task to improve performance on a target task through using knowledge learned from the source task in the learning process of the target task. Typically, a pre-trained model is then fine-tuned for the new target task with the sole objective of improving performance on the target task. As a result, the source task knowledge is overwritten with the new target task, and the fine-tuned model is no longer optimized for the previously trained source task. This effect of overwriting the learned knowledge in the model is known as catastrophic forgetting.

Scenarios arise in practice where preserving performance on the source task is crucial, especially under continual or lifelong learning objectives where a model is continuously optimized to new incoming tasks. In this chapter, we address the aforementioned scenario of lifelong learning and present a proposed solution to minimize the effect of catastrophic forgetting when fine-tuning pre-trained models to a stream of incoming target tasks.

5.3 Proposed Methods

In this work, we focused on leveraging LL to minimize catastrophic forgetting in neural networks trained on time-series data from sensing applications, such as human activity context recognition. Given that sensing data is abundantly living on edge devices, we assumed that data from previously trained tasks were no longer available due to limitations in data storage, computational capacity, and privacy concerns in storing personal data for long durations.

We propose a lifelong learning multitask autoencoder (LOMA) method to address the problem of fine-tuning a network previously trained on a rich, source task domain. This method enables training on new, target task(s) with limited labeled training data while minimizing the performance impact of source tasks where data are no longer accessible. The goal was to develop a method that can leverage knowledge from pre-trained source tasks to improve the performance of one or multiple new target tasks. However, in comparison to traditional FT methods, the challenge is to preserve the performance on old tasks where training data are no longer accessible. The assumption is that the network is pre-trained on a large, rich domain of old tasks and that the new tasks suffer from limited labeled training data in comparison to the rich source domain. In the inference phase, the network can accurately make predictions on the target tasks and has minimal catastrophic forgetting on the source tasks. Figure 5.1 presents the proposed LOMA model. Eq. 5.1 details the objective function of the proposed model, highlighting the play that each loss function has in Figure 5.1.

$$\begin{aligned} \Theta^* = \operatorname{argmin}_{\Theta^s, \Theta^o, \Theta^n} \Lambda_o \mathcal{L}_{kd}(Y^{o'}, \hat{Y}^{o'}) & \quad (5.1) \\ + (1 - \Lambda_o) \mathcal{L}_{ce}(Y^n, \hat{Y}^n) + \mathcal{L}_{rc}(X^n, \hat{X}^n) + \mathcal{R}_{\mathcal{M}\mathcal{T}\mathcal{L}}(\Theta) \end{aligned}$$

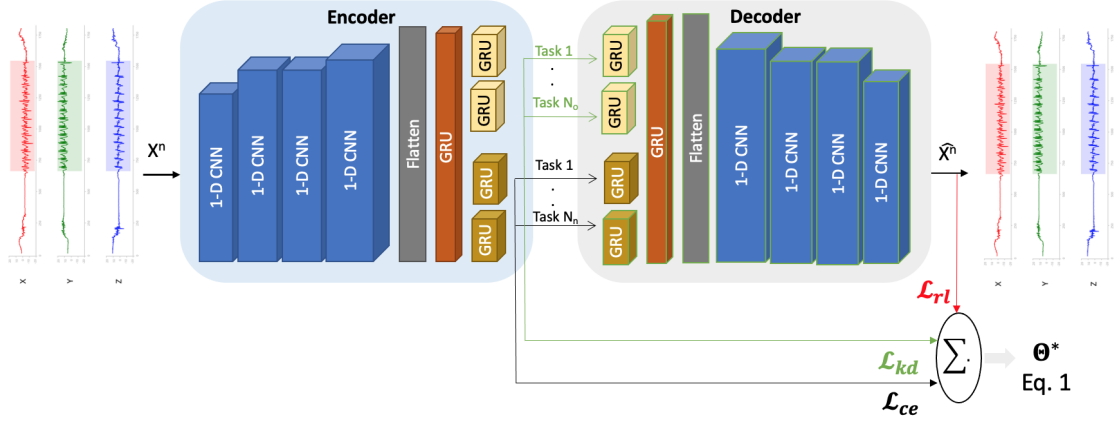


Figure 5.1: Diagram detailing the architecture and different losses involved in the learning stage of the proposed LOMA approach.

$$\Theta^* = \theta^{s*}, \theta^{o*}, \theta^{n*}$$

- o, n superscripts signify old source tasks and new target tasks.
- $\Theta_s^*, \Theta_o^*, \Theta_n^*$ are the optimal parameters for the shared layers, old task-specific layers, and new task-specific layers, respectively, after network training is completed.
- X^n and \hat{X}^n are the true input sequences fed into the encoder network and the reconstructed input sequences generated by the decoder network, respectively.
- Y^n and \hat{Y}^n are the true output labels and the recorded cross-entropy output labels of the new target tasks, respectively.
- $Y^{o'}$ are the recorded output labels of new tasks when generated from the old tasks' softmax layers from the pre-trained network, and $\hat{Y}^{o'}$ are the cross-entropy output labels of the new target tasks generated from the old task softmax layers during network fine-tuning.
- $\mathcal{L}_{kd}, \mathcal{L}_{ce}, \mathcal{L}_{rc}$ are knowledge distillation loss (described in Section 5.3.1), the cross-entropy classification loss (described in Section 5.3.2), and autoencoder reconstruction loss (described in Section 5.3.2), respectively.
- $\Lambda_o \in (0, 1)$ is a meta-parameter that defines the extent to which we favor preserving the loss on old tasks as opposed to new tasks in the objective function.
- $\mathcal{R}_{\mathcal{MTL}}$ is the MTL ℓ_2 -norm regularization on all network parameters Θ .

The proposed LOMA model tackles the following challenges:

1. Minimizes catastrophic forgetting on old tasks in comparison to prior state of the art, namely LwF [15] (Section 5.3.1).
2. Presents robust end-to-end representation learning between old and new tasks (Section 5.3.2).
3. Handles time-series sensing data that is stochastic and non-iid in nature (Section 5.3.3).

5.3.1 Catastrophic Forgetting Minimization

In order to accurately learn the new target task(s), we fine-tuned the LOMA network pre-trained on the old source tasks. Using the pre-trained network as a starting point boosts performance on new tasks, assuming that old and new tasks are related and that the new task(s) suffer from limited labeled data availability.

In addition to accurately learning the new task(s), we aimed to preserve performance or minimize forgetfulness on the old source tasks in the pre-trained model. Without access to source domain training data instances, it was impossible to re-train on old tasks and thus maintain equally sufficient performance on the source and target tasks. As such, a knowledge distillation (KD) loss was introduced when fine-tuning the network on the new tasks, following state-of-the-art LwF [15].

$$\mathcal{L}_{kd}(Y^{o'}, \hat{Y}^{o'}) = \sum_{t=1}^{N_n} \left[\ell(y_t^{o'}, \hat{y}_t^{o'}) + \sum_{t'=1}^{N_o} f(\beta_{t,t'}, \ell(y_{t'}^{o'}, \hat{y}_{t'}^{o'})) \right] \quad (5.2)$$

$$\ell(y_t^{o'}, \hat{y}_t^{o'}) = - \sum_{i=1}^{D_n} y_i^{o'} \log \hat{y}_i^{o'}$$

Eq. 5.2 presents the proposed KD loss where

- t reflects the task, where N_n is the total number of new target tasks and N_o is the total number of old source tasks.
- i reflects the data instance, where D_n is the total number of data instances for a given new target task t .
- $y_t^{o'}$ are the recorded output labels of the new task t data samples i generated from the old tasks' softmax layers from the original pre-trained network prior to any fine-tuning.
- $\hat{y}_t^{o'}$ are the output labels of the new task t data samples i generated from the old tasks' softmax layers during the network fine-tuning stage on the new tasks.
- ℓ reflects the loss function, which for our purposes on the classification task of human activity recognition is chosen to be the cross-entropy.

- $\beta_{t,t'}$ resembles the network parameters that are shared across a new task t and an old task t' in the shared layers of the network. This similarity parameter is zero for the network parameters that belong to task-specific layers.
- $f(\cdot)$ is the mapping function of the neural network architecture, particularly in this case, a CNN-GRU network layers for the old tasks' layers.

The main role of the KD loss during fine-tuning the pre-trained model for new tasks is to minimize the effect of catastrophic forgetting of old tasks. The KD loss, in Eq. 5.2, essentially computes a modified output label, $\hat{y}_t^{o'}$, for the new tasks from the old task softmax layers and constraints these modified output labels to remain within a close bound the recorded output labels, $y_t^{o'}$, from the original pre-trained network prior to any fine-tuning. Thus, the KD loss constraints the output labels to remain close and consequently does not allow the network parameters of the old task layers to undergo large deviations from the original pre-trained network, avoiding catastrophic forgetting of old tasks.

$$y_i^{o'} = \left(\frac{y_i^o}{\sum_j y_j^o} \right)^{1/T}, \hat{y}_i^{o'} = \left(\frac{\hat{y}_i^o}{\sum_j \hat{y}_j^o} \right)^{1/T} \quad (5.3)$$

$y_i^{o'}$ and $\hat{y}_i^{o'}$ are the modified output labels from old task softmax layers for data instance i summed over j , which iterates across all data instances. These modifications alter the traditional cross-entropy loss to increase the importance of smaller probabilities, thus minimizing the forgetfulness of the network. A value of T greater than 1 was recommended by Hinton, et al. [84] as it will increase the importance of smaller probabilities. We set $T = 2$ for our experiments as suggested by [15].

5.3.2 Robustness with End-to-End Representation Learning

We propose a multitask autoencoder architecture that is capable of learning multiple tasks simultaneously while learning end-to-end task representations. We describe the multitask learning and autoencoder architectures in the sections below.

Multitask Learning

At its core, the LOMA model constitutes a base model, which is the neural network topology chosen to learn any of our desired tasks. In this study, the base model was a multitask learning network that learns multiple tasks simultaneously through hard-parameter sharing [85]. The model is composed of a hierarchical structure of a convolutional neural network (CNN) followed by a gated recurrent unit (GRU) and shown in Figure 5.1. The CNN layers and first GRU layers of the encoder are shared across tasks while the second GRU and softmax layers are task-specific to their corresponding task in the encoder. In contrary, the decoder architecture defines task-specific layers as the first GRU layer while the second GRU and remaining CNN layers are shared across all tasks.

For the purpose of this work, the multitask network was composed of multiple heads (or task-specific layers) for both old and new tasks. \mathcal{L}_{ce} is the traditional cross-entropy loss function defined for multilabel classification, and aims to minimize the classification error between target labels, Y_n , and network output predictions, \hat{Y}_n .

$$\mathcal{L}_{ce}(Y^n, \hat{Y}^n) = \sum_{t=1}^{N_n} \left[\ell(y_t^n, \hat{y}_t^n) + \sum_{t'=1}^{N_n} g(\alpha_{t,t'}, \ell(y_{t'}^n, \hat{y}_{t'}^n)) \right] \quad (5.4)$$

$$\ell(y_t^n, \hat{y}_t^n) = - \sum_{i=1}^{D_n} y_i^n \log \hat{y}_i^n$$

Eq. 5.4 presents the proposed LOMA cross-entropy classification loss where

- \hat{y}_t^n, y_t^n are the predicted and true output labels for a given new task t , respectively.
- ℓ reflects the loss function, which for our purposes on the classification task of human activity recognition is chosen to be the cross-entropy.
- $\alpha_{t,t'}$ resembles the network parameters that are shared across a new task t and another new task t' in the shared layers of the network. This similarity parameter is zero for the network parameters that belong to task-specific layers.
- $g(\cdot)$ is the mapping function of the neural network architecture, particularly in this case, a CNN-GRU network layers for the new task(s)' layers.

Given that the old task data were no longer accessible during training of new tasks, traditional multitask learning was no longer possible. This is where the knowledge distillation of the LOMA model comes into play in order to preserve performance on old tasks as described in Section 5.3.1.

Autoencoder

The proposed autoencoder architecture enforces improvements in generalizability of the network which is expected to help mitigate the effects of negative transfer that arises when tasks are sharing information during training. Through the shared encoder and decoder layers, the encoded representation and reconstruction is being learned and shared across the various tasks, enforcing representations to be shared and restricting the occurrence of large distribution shifts that potentially arise from the time-series data having distribution shifts in the features of different tasks, or “covariate shift”.

$$\mathcal{L}_{rc}(X^n, \hat{X}^n) = - \sum_{i=1}^{D_n} \frac{1}{2} (\hat{x}_i^n - x_i^n)^2 - \lambda \sum_k KL(\rho || \hat{\rho}_k) \quad (5.5)$$

$$\sum_k KL(\rho || \hat{\rho}_k) = \rho \log \frac{\rho}{\hat{\rho}_k} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_k}$$

Eq. 5.5 defines the reconstruction (RC) loss of the autoencoder along with a Kullback-Leibler (KL) divergence penalty for sparsity enforcement [86]. The KL-divergence penalty of ρ , which is a sparsity parameter that is typically defined close to 0.05 [86], constraints the estimated $\hat{\rho}$ to remain as close as possible to ρ across

all hidden units of the autoencoder - thus, enforcing the sparsity of the encoded representations.

$$\hat{\rho}_k = \frac{1}{D_n} \sum_{i=1}^{D_n} [a_k^{(l)}(x^{(i)})]$$

where i is the number of samples, k is the hidden unit, l is the network layer, D_n is the total number of samples, and x is a given input data sample for which the sparse reconstruction is estimated.

5.3.3 *Time-series Data Architecture*

The network architecture is composed of a hierarchical structure of a convolutional neural network (CNN) layer followed by gated recurrent unit (GRU) layers. The hierarchy of a CNN followed by a GRU is an architecture design commonly used in literature for time-series problems [67], [70]–[72]. Specifically, the multitask architecture design was adopted following state of art work on multitask time-series classification problems in the work of Mahmoud et al. [38]. The CNN layers are capable of extracting feature representation from the time-series sequences through one-dimensional convolution operations. The GRU layers are capable of capturing short- and long-term dependency in the time-series sequences such that the stochastic, time-varying nature of the data is sufficiently learned.

The layer hyperparameters are specified in Table 5.1. The time-series sequences are segmented into fixed size windows and fed into the network. Each window is of the size of 128 samples, with a 50% overlap existing across consecutive windows.

5.4 Data & Experimental Setup

We evaluated our proposed LOMA approach under the supervised task of human activity recognition from wearable and mobile devices. Activity recognition was chosen to evaluate the presented work for sensing domain applications as it has been widely studied and provides a large array of well-established benchmark datasets [87]. As such, it serves to provide a good illustration of the potential of lifelong learning for time-series data.

5.4.1 *Datasets*

In our study, we evaluated the proposed LOMA multi-objective approach under 4 benchmark HAR datasets: ALKAN [12], DAS [18], OPPORTUNITY [19], and UCI HAR datasets [20].

ALKAN [12] is a large-scale HAR dataset that is collected in a natural environment from user smart phones while they go about their daily lives. The data is collected from an accelerometer sensor which samples time stamps at 20 Hz. The activity labels generated from users are mainly locomotion activities such as walking, running, and riding a bus. The dataset is collected from 20 user subjects which are treated as separate tasks in our experiment.

The Daily and Sports Activities (DAS) [18] dataset is composed of a wide set of 19 activities, across locomotion and high-level activities, which are performed by 8 user subjects. The dataset is collected using 5 sensor devices located on a user’s torso, right and left arms, and right and left legs. Each sensing device samples data at 25 Hz from a 3D accelerometer, gyroscope, and magnetometer.

OPPORTUNITY [19] was collected in a controlled experimental setup using numerous on-body sensors and a sensor-rich environment. The time-series data were collected from the sensor modalities at a 30 Hz frequency. The activity labels include locomotion activities (e.g., walking, running, sitting, and lying down) and gesture recognition (e.g., close fridge). For our experiments, we focused on both of these label domains. The 4 user subjects from OPPORTUNITY represent different tasks in our experiments.

UCI HAR [20] is a collection of results from 30 user subjects who performed six basic locomotion activities, including walking, walking upstairs, walking downstairs, sitting, standing, and laying down. Data samples were collected from one smart-phone attached to the waist of each subject, and thus corresponds to sensing data from accelerometer and gyroscope sensors. The time-series data were collected from the sensor modalities at a 50 Hz frequency.

Under different experimental setups, we treat different datasets as the source or target domains interchangeably to evaluate the proposed LOMA model’s performance under various settings. Details of data count statistics, data splits, and label distributions can be found in Appendix A.

5.4.2 *Experimental Setup*

Implementation Details

The proposed network was implemented in Python using Google’s Tensorflow. The multitask network is trained using alternative training. This is an iterative training process where at every iteration a batch is selected from task t to update the shared layers and task-specific layers of task t in the network. In the next iteration, task $t + 1$ batch is fed into the network, which again updates the shared layers and task-specific layers of task $t + 1$, so on so forth.

For experimental evaluation, we used the accuracy of human activity recognition tasks, where the goal was to classify the correct activity label per window frame of the input time-series sequence. Accuracy was defined as the number of correctly classified window frames divided by the total number of frames being evaluated. Table 5.1 presents the network architecture hyperparameters. The network was trained using 128 batch size, 30 epochs, and 0.2 dropout across the GRU layers of the encoder and decoder networks. The network hyperparameters, including batch size, dropout rate, convolutional filter count, and GRU cells were chosen following a grid search. The hyperparameter values with the best attained accuracy on the validation set split from the grid range were selected for the final model evaluation on the test set splits. The number of epochs for training were set using Tensorflow’s early stopping criterion.

Baselines

We compared the proposed approach against (1) a variant of prior state of art on Learning without Forgetting (LwF) [15] adopted for time-series neural network architecture, (2) a traditional fine-tuning (FT) baseline, (3) a multitask learning (MTL) baseline, and (4) a single-task learning (STL) baseline. State of the art LwF is the encoder network with equal weighting given to both the knowledge distillation and cross-entropy losses of Equation 1, where both are set to 1. FT is the baseline network where traditional transfer learning is used to fine-tune the pre-trained model with $\Lambda_o = 0$ in Eq. 5.1 on old tasks. STL is the baseline network trained only on new tasks. Finally, MTL is the baseline network where all old and new tasks are learned simultaneously. Here, the data for old tasks were available, and so naturally, MTL was expected to serve as an upper baseline for the proposed approach. Furthermore, we run an ablation analysis to evaluate the impact of the different losses in the multi-objective learning function proposed in LOMA.

Table 5.1: Architecture Hyperparameters

Layers	Hyperparameters	Values
CNN Layer 1	Filter size	32
	Kernel size	(1,3)
	Stride	1
CNN Layers 2, 3	Filter size	64
	Kernel size	(1,3)
	Stride	1
CNN Layer 4	Filter size	72
	Kernel size	(1,3)
	Stride	1
GRU Layers 1, 2	Cells	128

5.5 Results & Discussion

5.5.1 *Single New Task*

Comparison against State-of-the-art LWF

In the first setup of the experiment, we introduced one of four new tasks from OPPORTUNITY with locomotion activity labels into the pre-trained model independently. LwF mitigated catastrophic forgetting on old tasks by showing superior performance on these tasks compared to FT, where preserving performance on old tasks was neglected (Table 5.2). In comparison to LwF, LOMA performed even better to mitigate catastrophic forgetting of old tasks, with comparable to improved performance on new tasks. Furthermore, LOMA was able to better maintain performance on old tasks. This behavior is likely attributed to the introduction of the decoder network, which forces new task representations to remain bounded by the distilled output of old task network parameters.

In the second setup of the experiment, we introduced one of four new tasks from OPPORTUNITY with gesture activity labels into the pre-trained model indepen-

Table 5.2: Experiments evaluated on 30 old tasks from UCI HAR and 4 new tasks from OPPORTUNITY. Old and new tasks have similar labels of basic locomotion activities. Results are shown in accuracy (%).

	UCI	UCI <i>old</i>	User 1 <i>new</i>	UCI <i>old</i>	User 2 <i>new</i>	UCI <i>old</i>	User 3 <i>new</i>	UCI <i>old</i>	User 4 <i>new</i>
LOMA	-	81.32	82.11	84.09	82.22	77.3	81.69	82.5	73.20
LwF	-	79.68	82.88	82.21	81.60	75.14	81.34	80.56	74.03
FT	-	78.89	83.67	81.18	84.3	74.52	80.90	79.33	75.76
MTL	88.71	87.33	81.45	88.28	84.60	85.62	79.16	86.88	76.08
STL	-	-	80.76	-	83.08	-	79.23	-	73.65

Table 5.3: Experiments evaluated on 30 old tasks from UCI HAR and 4 new tasks from OPPORTUNITY. Old and new tasks have different labels of basic locomotion activities and gesture recognition activities, respectively. Results are shown in accuracy (%).

	UCI	UCI <i>old</i>	User 1 <i>new</i>	UCI <i>old</i>	User 2 <i>new</i>	UCI <i>old</i>	User 3 <i>new</i>	UCI <i>old</i>	User 4 <i>new</i>
LOMA	-	74.66	67.11	73.90	71.11	71.36	66.75	70.15	71.23
LwF	-	69.43	66.78	71.04	72.86	65.53	66.21	67.70	72.40
FT	-	71.8	66.09	72.67	74.32	70.1	68.26	67.58	70.88
MTL	88.71	81.23	67.38	80.67	71.64	79.12	68.44	74.57	67.02
STL	-	-	71.23	-	76.82	-	71.66	-	73.19

dently. In this setup, a label shift existed between the old and new tasks, where labels of old and new tasks belonged to different distributions. We consistently witnessed improved preservation of old task performance in LOMA compared to LwF (Table 5.3). The interesting distinction with regards to label shifts between old and new tasks was in the performance of FT compared to LOMA and LwF. We observed superior performance of LOMA and LwF on users 1 and 4 in comparison to FT.

Comparison against the FT Baseline

Table 5.2 shows that FT performed better on new target tasks with no label shift, given it was prioritizing the learning of new tasks and disregarding old pre-trained ones. The advantage of LOMA is in its ability to significantly minimize catastrophic forgetting of old tasks compared to FT, while achieving relatively equivalent performance on new tasks. In comparison, where new tasks exhibit a label shift in comparison to old tasks, we observed improved performance with LOMA compared to FT on two of the newly introduced tasks, users 1 and 4 (Table 5.3).

Comparison against Single-task Learning (STL) Baseline

We also compared LOMA against STL performance on new tasks alone trained in an encoder network from random weight initialization (Tables 5.2 and 5.3). Here the encoder network had not been pre-trained on old tasks. The results show STL acted as an upper bound on the performance of new tasks given that LOMA sacrificed performance on new tasks to preserve old ones. We observed that FT exhibited higher performance than STL in Table in the absence of label shifts on new tasks (Table 5.2). This was expected since FT leveraged pre-trained knowledge from old tasks to improve the learning of new tasks, especially given that the old tasks came from a richer domain in comparison to the new ones.

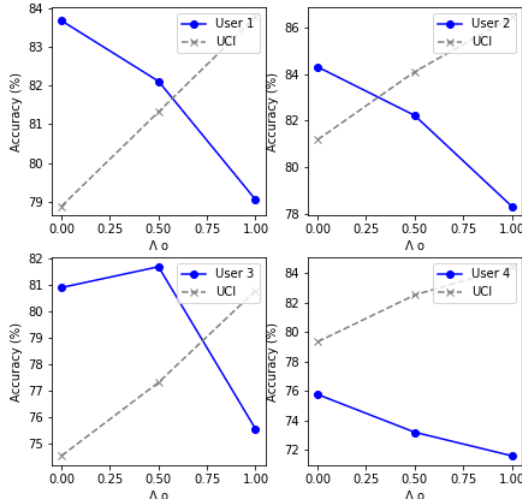


Figure 5.2: Comparing LOMA performance when Λ_o is varied between $[0.05, 0.95]$ on old tasks in the objective function of Eq. 5.1

STL exhibited superior performance to FT when a label shift existed between old and new tasks (Table 5.3). FT a new task with a label shift seemed to negatively impact performance when compared to the STL approach. STL was actually seen as an upper bound for new tasks with label shift in all methods evaluated, including LOMA, LwF, and MTL.

Comparison against Multitask Learning (MTL) Baseline

In the first experimental setup of Table 5.2, MTL is shown as the upper boundary in performance on the old tasks, which was expected given that the model had full access to the old task data and was trained simultaneously with the new tasks.

Interestingly, when a label shift existed between old and new tasks in the second experimental setup in Table 5.3, MTL was negatively impacted by the introduction of a task with a label shift. This is not unlikely given that MTL models are expected to showcase their strength when tasks learned simultaneously do not exhibit large distribution shifts.

To test the performance of our proposed LOMA method with tasks of varying importance, we varied Λ_o between 0.05 and 0.95 in Eq. 5.1 to increase the weight of the old task distillation output labels compared to new tasks being introduced into the network. The results are shown in Figure 5.2.

When $\Lambda_o = 0.05$, LOMA was closely resembling the FT setup, where minimal weight was given to preserve old tasks. Here we observed the lowest performance on pre-trained UCI old tasks and the highest performance on new tasks for users 1, 2, 3, and 4. When $\Lambda_o = 0.5$, equal weight was given to the learning of old and new tasks. In this case, the best combined performance across old UCI and new tasks occurred for users 1, 2, 3, and 4. On the other end of the spectrum, when $\Lambda_o = 0.95$, minimal weight was given to the learning of new tasks. Here the highest

Table 5.4: Training combinations of new tasks from OPPORTUNITY simultaneously.

UCI	User 1	User 2	User 3	User 4	AVG
81.32	82.11	-	-	-	81.72
80.78	83.67	80.75	-	-	81.73
78.61	81.45	79.04	78.56	-	79.41
77.33	81.25	77.69	77.73	74.23	77.65

performance was observed on old task UCI and the lowest was seen on new tasks for users 1, 2, 3, and 4.

5.5.2 *Multiple New Tasks*

In the next set of experiments, we evaluated how the LOMA model would perform when multiple new tasks are introduced. Here, we conducted two experimental setups. The first setup involved training on multiple new tasks simultaneously. That is, tasks were fed to the model for training in parallel. In the second setup, we trained on the incoming new tasks sequentially. Note the experimental results were run on OPPORTUNITY user subjects with locomotion labels (i.e., no label shift between old and new tasks).

Simultaneous Training of New Tasks

We began by introducing all four new tasks into the model in parallel, where target tasks were trained simultaneously as seen in traditional MTL setups. In practice, this setup assumes a waiting buffer period before all target tasks are available.

As more target tasks are trained simultaneously, the average performance of the LOMA method dropped (Table 5.4). This outcome may be explained by the model’s generalization overpowering its personalization aspect, where incoming tasks have larger distribution shifts relative to the small task count.

We examined user 1, whose performance improved when trained in parallel with user 2. However, the performance of user 2 dropped when trained simultaneously with users 1 and 3. This outcome raises the question of when tasks would experience a positive versus negative transfer of shared knowledge in the network. We present two possible factors that may impact performance. First, noisy data might be having a negative effect on the training of neighboring task parameters. For example, a particular new user could have so many training samples that adding noisy data (originating from users with distribution shifts) decreases performance.

Second, similar to MTL, learning tasks simultaneously is expected to improve performance only if the training data introduces positive shared knowledge, particularly when a given task suffers from a limited training data count.

Sequential Training of New Tasks

Next, we examined the setup where multiple new target tasks are introduced into the model sequentially. This setup was expected to be impacted by larger catastrophic forgetting compared to simultaneous learning since the network becomes more susceptible to forgetting previously trained information with every new trained task. This outcome was confirmed by the results presented in Tables 5.5

Table 5.5: Training all target tasks from OPPORTUNITY in sequence. The order of tasks as introduced to the model is: 1,2,3,4.

UCI	User 1	User 2	User 3	User 4	AVG
81.32	82.11	-	-	-	81.72
78.77	80.85	82.65	-	-	80.75
72.40	78.45	79.21	81.32	-	77.85
68.33	72.67	71.82	78.41	76.55	73.56

Table 5.6: Training all target tasks from OPPORTUNITY in sequence. The order of tasks as introduced to the model is: 4,3,2,1.

UCI	User 1	User 2	User 3	User 4	AVG
82.5	-	-	-	73.20	77.85
74.34	-	-	78.45	69.78	74.19
70.2	-	80.05	75.67	67.33	73.31
65.54	80.86	72.66	71.23	60.10	70.08

and 5.6. We observed that, as more tasks were added to the model sequentially, the relevance of old tasks decayed exponentially. In this case, controlling the knob of Λ_o may have an important impact on preserving old tasks.

Nevertheless, our data demonstrate that the performance of user 4 was better following the introduction of users 1, 2, and 3 into the LOMA model compared to when user 4 was introduced alone into the pre-trained model (Table 5.5). This is most likely due to the feature distribution of user 4 being more closely related to the new OPPORTUNITY user subjects as opposed to subjects from the UCI HAR representing old tasks.

To further investigate the sequential learning setup, we examined whether the order in which tasks are presented to the model impacts performance. Results in Tables 5.5 and 5.6 demonstrate that the average performance of the model varied according to the order of new users introduced (e.g., 1-2-3-4 versus 4-3-2-1). For example, user 2 exhibited a performance of 82.65% when it followed the training of user 1 (e.g., 1-2-3-4). However, when the order of tasks was reversed (e.g., 4-3-2-1), the performance of user 2 dropped by -2.6% to 80.05%. This difference in performance was also observed for the other tasks. This outcome raises interesting questions about how tasks impact one another in the model.

Moreover, these results highlight the importance of understanding the optimal order in which tasks should be introduced to maximize performance. To examine this, we performed an experimental grid search across the four new tasks with 24 combinations to determine the optimal order in which tasks should be introduced to the model. The optimal sequence of new tasks was found to be: 2-1-3-4. This sequence exhibited (1) the highest average accuracy of the model after the introduction of all tasks, and (2) the lowest drop in average accuracy after each new task is trained. Sequence 2-1-3-4 had an average accuracy of 76.23% and an average accuracy drop of 5.4% from beginning to end of the training sequence.

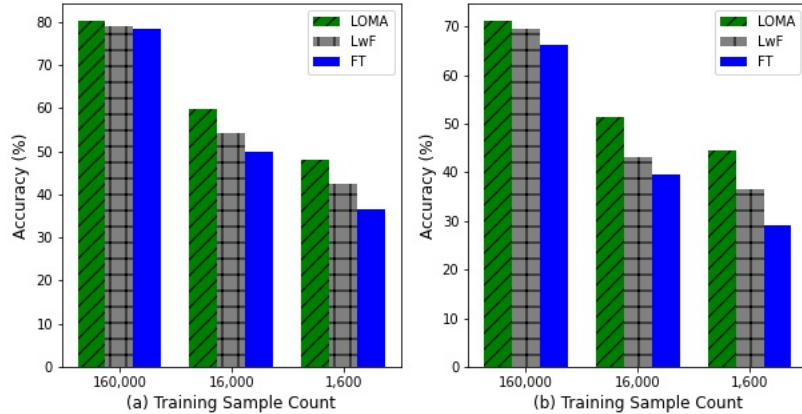


Figure 5.3: The plots present average performance across the 5 OPP users for which the network is fine-tuned after being pre-trained with ALKAN 20 users. The training data size of ALKAN is reduced in orders of magnitude to showcase robustness of the methods to limited pre-trained data. (a) represents fine-tuning on OPP with no label shift, and (b) represents fine-tuning on OPP with label shift.

These experimental results demonstrate that the order in which tasks are learned matters. To further validate how the order impacts model performance, future studies can be performed to examine how the model is impacted by the error bounds of a given task [88]. The preliminary hypothesis is that a task with 'noisier' data or that presents a larger distribution shift into the model would lead to lower performance. This lower performance would likely be a result of lower generalization capability of the model since a larger distribution shift would force the model to tailor its parameters towards the newly learned shift. This effect is more prominent when tasks are learned sequentially since, at the introduction of every new task, the LOMA objective function (see Eq. 5.1) gives equal weighting to the new task and all other old tasks. Finally, it would be very interesting for future studies to investigate how the objective function can compute an order of tasks that produces optimal performance in terms of a minimized average error bound.

Robustness against Limited Training Data

The results presented in Figure 5.3 highlight LOMA's robustness in learning new tasks when pre-trained with limited labeled data samples. For these experiments, LOMA was pre-trained on ALKAN with 20 user subjects as the old tasks and fine-tuned for OPPORTUNITY with 4 user subjects (a) with no label shift and (b) with an exhibited label shift. The results in Figure 5.3 are averaged across the 4 user subjects of OPPORUNITY.

Results are presented for 3 experimental setups where pre-training happens on 160,000, 16,000, and 1,600 data instances from the ALKAN old tasks to showcase performance of LOMA and baseline methods under reductions in orders of magnitude of the training data sizes. The figure demonstrates the robustness of the proposed LOMA method under reduction in training data sizes in comparison to prior state of art loss functions LwF and traditional FT methods. With LwF showing

Table 5.7: Summary of ablation analysis run on 5 experimental setups. Results are shown in average accuracy (%) across all tasks.

Loss	UCI	OPP	UCI	ALKAN	ALKAN	UCI	ALKAN	DAS	OPP	
	<i>old(30)</i>	<i>new(4)</i>	<i>old(30)</i>	<i>new(5)</i>	<i>old(20)</i>	<i>new(5)</i>	<i>old(20)</i>	<i>new(8)</i>	<i>old(20)</i>	<i>new(4)</i>
+K+R	81.33	80.30	83.20	79.45	60.21	87.89	57.81	84.32	56.67	80.15
-R	79.39	79.96	81.47	76.57	57.5	86.43	55.04	81.56	57.04	78.96
-K-R	78.48	81.15	79.24	79.87	56.75	88.93	52.67	87.23	54.61	78.32

superior ability to handle limited data to traditional FT, the proposed combination of knowledge distillation and reconstruction losses outperform state of art objectives in ability to handle limited training data, even under the case of an existing label shift in the label categories of the source and target domain.

5.5.3 Ablation Analysis

We conducted an ablation analysis to study the impact of the proposed knowledge distillation and reconstruction loss functions presented in our proposed multi-objective LOMA approach. The columns of Table 5.7 present the different experimental setups that highlight the 'Source - Target' datasets for that experiment. Each experiment column reports performance on the old and new tasks, where the number shown in the brackets indicates the number of tasks (or user subjects) for each source and target dataset. The results in the table are reflecting the average accuracy performance in (%) on all old or new tasks, respectively.

The presented results in the table cover evaluation of the following setups:

1. All losses (K+R): This case represents the full proposed LOMA multi-objective loss function with both knowledge distillation (KD) and reconstruction (RC) losses.
2. No reconstruction loss (-R): The multi-objective loss function excluding the RC loss.
3. No reconstruction and knowledge distillation (-K-R): The multi-objective loss function excluding the RC and KD losses.

Table 5.7 summarizes 5 experimental setups that show the average accuracy performance of the ablation analysis across all user subjects for old and new tasks under different datasets. The results show consistent improvement of the proposed multi-objective loss function, combining the KD and RC losses, over both old and new tasks in comparison to eliminating the RC and KD losses. To be more specific, when exploring the comparison of the proposed multi-objective with both KD and RC (K+R) against eliminating both losses (-K-R), we notice (1) a significant average improvement of +3.494% on preserving old tasks' performance across the 5 experiments conducted on 4 different datasets and setups of source-target pairs, and (2) a minimal loss of -0.678% in average performance on new tasks. This minimal loss is expected and labeled as "minimal" since the (-K-R) setup reflects the

traditional fine-tuning setup which is considered to serve as an upper bound for our proposed LOMA method on the old tasks.

Finally, when comparing the performance of the proposed multi-objective loss (K+R) with case where the RC loss is eliminated (-R), we witness a consistent improvement of +1.756% and +1.726% on old and new tasks, respectively. This improvement is attributed to the presence of the reconstruction loss, which was proposed as a mechanism to eliminate negative transfer across new tasks being learned (shown successfully in the improvement witnessed on new tasks). However, in addition to that, we can see that the RC loss has played a role in improving the ability to preserve performance on old tasks. This can be attributed to the decoder shared layers enforcing more shared representations across the old and new tasks, and thus, further minimizing deviations in the learned parameters of old tasks.

In summary, the ablation study presented the following takeaways: (1) consistent improvement of the proposed multi-objective loss function, combining the KD and RC losses, over both old and new tasks, (2) a significant average improvement of +3.494% on preserving old tasks' performance across 5 experimental setups conducted on 4 different datasets, (3) a minimal loss of -0.678% in average performance on new tasks when compared to the upper bound of traditional fine-tuning where both KD and RC losses are eliminated, and finally (4) the RC loss, which was aimed at improving performance on new tasks by mitigating effects of negative transfer through improved generalization, was shown to further contribute to enhanced performance on old tasks preservability.

5.6 Broader Applications

The presented methods of this chapter can be directly extended to different types of time-series problems, such as time-series forecasting applications, by using the appropriate choice of loss function and output layer activation function in the fine-tuning of the pre-trained network and the new output branches introduced per new target task, respectively.

CHAPTER 6

MEASURES FOR TRANSFERABILITY ESTIMATION OF PRE-TRAINED MODELS

6.1 Overview

In this chapter, we address the problem of estimating task transferability between a source pre-trained model and a target task. Once again, we follow the assumption that the source task data is no longer accessible. The goal is to define a measure of task transferability that serves as a priori knowledge of the success of transfer from a source pre-trained model to a target task and does not require expensive optimization.

The contributions of Chapter 6 are as follows:

1. A new measure, iLEEP, that learns from an attention-based network the optimal combination of target representations for transferability estimation.
2. An interpretable transferability estimation method that learns an intuitive mapping of the relationship between source and target tasks.

We next present the proposed methods and evaluate the methods on two benchmark computer vision datasets: CIFAR10 [26] and Domainnet [27], where CIFAR10 contains camera-captured images and Domainnet presents images collected from varying input domains including paintings, sketches, clipart, and infographics. We show that our proposed iLEEP measure improves on prior state-of-the-art and presents an interpretable, intuitive definition of transferability between a source pre-trained model and target task.

6.2 Problem Definition

In a transfer learning setting, we are typically interested in using a source task to improve performance on a target task through using knowledge learned from the source task in the learning process of the target task. The source task is a

task, which has an abundant source of labeled training data samples, to be used to help improve model training on our target task, which suffers from limited labeled training data samples. Moreover, under our target problem scenario in this chapter, we are dealing with a single source task and a single target task, where the source task dataset is assumed to be no longer available. This may be due to the data being lost, limitations in storage resources, or constraints related to privacy or security.

The target task is represented by dataset \mathcal{D} , which contains input sample and output label pairs such that $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$. The input samples belong to domain $\mathcal{X}_T = \mathbb{R}^N$ and can be of image, text, or time-series data types that in return can be mapped to and represented by an N -dimensional vector. The output labels belong to a finite domain \mathcal{Y} .

The source task maps a given input domain $\mathcal{X}_S = \mathbb{R}^N$ to a finite output domain \mathcal{Z} . In our settings, the input-output sample pairs for the source task are no longer available, and thus, the source task can only be represented by a pre-trained model θ , where θ represents the optimized parameter set of the pre-trained model that maps the input domain to the output domain, \mathcal{X}_S and \mathcal{Z} , respectively.

6.3 Proposed Methods

In this chapter, we focused on studying the underlying relationship between a pre-trained model on a source task and a dataset of a new target task, for which we wish to optimize the pre-trained model. Previous work has explored extracting feature representations of the target task from the pre-trained model as a reflection of the old source task’s closeness to the given target task. We elaborate on the prior state-of-the-art techniques in Section 6.3.1 to serve as a background to the discussion of our proposed methods.

6.3.1 Log Expected Empirical Prediction Measure

C. Nguyen et al. [17] first introduced the Log Expected Empirical Prediction (LEEP) score, which serves as a measure to estimate the transferability between a pre-trained source model and a target dataset. The LEEP score does not require expensive computational fine-tuning of the pre-trained model and serves as an apriori assessment of the success of transfer between the pre-trained model and the target dataset.

The LEEP measure is defined in Eq. 6.1, where $\hat{P}(y_i|z)$ is the empirical conditional distribution of the target outputs \mathcal{Y} given the source outputs \mathcal{Z} and $\theta(x_i)_z$ is the probability of a given source label $z \in \mathcal{Z}$ following the output distribution of passing the target samples through the source pre-trained model, $\theta(x_i)$.

$$\text{LEEP}(\theta, \mathcal{D}) = \frac{1}{n} \sum_{i=1}^n \log \left(\sum_{z \in \mathcal{Z}} \hat{P}(y_i|z) \theta(x_i)_z \right) \quad (6.1)$$

The LEEP measure can be computed over three steps.

Step 1: Compute the dummy label distribution of the target dataset $\theta(x_i)$. To compute the dummy label distribution, we pass the input target instances

x_i through θ to generate the distribution of outputs over \mathcal{Z} , the source data output domain, that is $\theta(x_i)$. It is called 'dummy' label distribution since it does not represent the true target outputs and is likely entirely meaningless to the context of the target outputs. $\theta(x_i)_z$ is then the probability of the source label given the dummy label distribution $\theta(x_i)$. The dimensions of $\theta(x_i)_z$ are then $(N_n \times N_z)$, where N_n is the number of target training samples that are passed through the source pre-trained model θ and N_z is the number of units in the last output layer of the source pre-trained model.

Step 2: Compute the empirical conditional distribution $\hat{P}(y|z)$. To compute the empirical conditional distribution for all pairs of $(y, z) \in \mathcal{Y} \times \mathcal{Z}$, we first compute the empirical joint probability, $\hat{P}(y, z)$. We then use the empirical joint probability to compute the empirical marginal probability $\hat{P}(z)$. Finally, we use the results of the empirical joint and marginal distributions to compute the empirical conditional probability $\hat{P}(y|z)$ in Eq. 6.2.

$$\hat{P}(y|z) = \frac{\hat{P}(y, z)}{\hat{P}(z)}, \quad (6.2)$$

$$\hat{P}(y, z) = \frac{1}{n} \sum_{i:y_i=y} \theta(x_i)_z,$$

$$\hat{P}(z) = \sum_{y \in \mathcal{Y}} \hat{P}(y, z) = \frac{1}{n} \sum_{i=1}^n \theta(x_i)_z.$$

Step 3: Compute the LEEP measure using $\theta(x_i)$ and $\hat{P}(y_i|z)$. Finally, we compute the LEEP measure given a source pre-trained model and a target dataset by plugging in the computed dummy label distribution, $\theta(x_i)$, from step 1 and the empirical conditional distribution, $\hat{P}(y_i|z)$, from step 2 into Eq. 6.1.

LEEP has shown great progress in successfully estimating the transferability of a pre-trained model to a target task of interest without the need for expensive optimization. However, LEEP suffered from multiple limitations such as (1) being limited to tasks of a classification nature and (2) being prone to overfitting as the dummy label distribution $\theta(x_i)$ is extracted from the softmax layer of the pre-trained network that has been optimized for the source task.

Y. Li et al. [25] proposed an improvement on LEEP, which they referred to as Gaussian LEEP (or \mathcal{N} LEEP), to overcome the two aforementioned limitations. \mathcal{N} LEEP advances LEEP by replacing the dummy label distribution $\theta(x_i)_z$ from the softmax output of the pre-trained model by the posterior probability distribution, $\hat{P}(v|x)$, computed across Gaussian components $v \in \mathcal{V}$. \mathcal{N} LEEP extracts the feature representation of a forward pass of the target samples from the penultimate layer of the source pre-trained network, that is, the layer before the output layer. The extracted representation is then passed through Principal Component Analysis (PCA) for a low-dimensional vector representation of the target input samples x . Moreover, PCA also allows mapping all the layer representations into a unified vector dimension. The low-dimensional vector representations are then fitted to a

Gaussian Mixture Model (GMM) $P(s) = \sum_{v \in \mathcal{V}} \pi_v \mathcal{N}(s | \mu_v, \sigma_v)$, where s is the target training dataset such that $s_{i_i}^n$. Moreover, \mathcal{V} represents the clusters of Gaussian components and π_v are the learned Gaussian mixture weights. The posterior probability distribution can be easily mapped to reflect the true target input samples, such that $P(v|x) = P(v|s) \propto \pi_v \mathcal{N}(s | \mu_v, \sigma_v)$.

The improved \mathcal{N} LEEP measure is represented in Eq. 6.3, where $P(v|x)$ has replaced the dummy label distribution, $\theta(x_i)$.

$$\begin{aligned} \mathcal{N}\text{LEEP}(\theta, \mathcal{D}) &= \frac{1}{n} \sum_{i=1}^n \log \left(\sum_{z \in \mathcal{Z}} \hat{P}(y_i|z)' P(v|x) \right), \text{ where} & (6.3) \\ \hat{P}(y|z)' &= \frac{\hat{P}(y, z)'}{\hat{P}(z)'}, \\ \hat{P}(y, z)' &= \frac{1}{n} \sum_{i: v_i=v} P(v_i|x), \\ \hat{P}(z)' &= \sum_{y \in \mathcal{Y}} \hat{P}(y, z)' = \frac{1}{n} \sum_{i=1}^n P(v_i|x). \end{aligned}$$

As a result, \mathcal{N} LEEP improves on LEEP by eliminating the reliance on the softmax classification output layer, making the \mathcal{N} LEEP measure applicable to any type of task including regression tasks and unsupervised tasks. Moreover, by evaluating the transferability measure with the posterior probability of the GMM trained on the target task (as opposed to the softmax layer trained on the source task), \mathcal{N} LEEP presents a more reliable measure of transferability for the target task.

Nevertheless, both LEEP and \mathcal{N} LEEP are restricted to examining the final resulting model representation of the source pre-trained model and do not account for the varying representations extracted at shallow versus deep layers in the network, which could bring forth added value when exploring the transferability relationship between a source and target task.

6.3.2 *Transferability with Multi-granular Representation Extraction*

Deep neural networks have been shown to serve as effective feature extractors. Prior work has demonstrated that the depth of the neural layers presents different stages of granularity in the features extracted. In particular, it has been shown that shallow layers tend to extract low granularity features while deeper layers extract high granularity features. For example, as shown in Figure 6.1, if we are passing the image of a human face, shallow layers would extract finer features such as edges and shades, mid layers would extract higher level facial features such as eyes and nose, and deep layers would extract highest level features such as the target object of a full face.

Motivated by the multi-granular representation extracted, we study the effectiveness of looking at the softmax or penultimate layer representations as opposed to all other layers in a pre-trained model for identifying the transferability of the

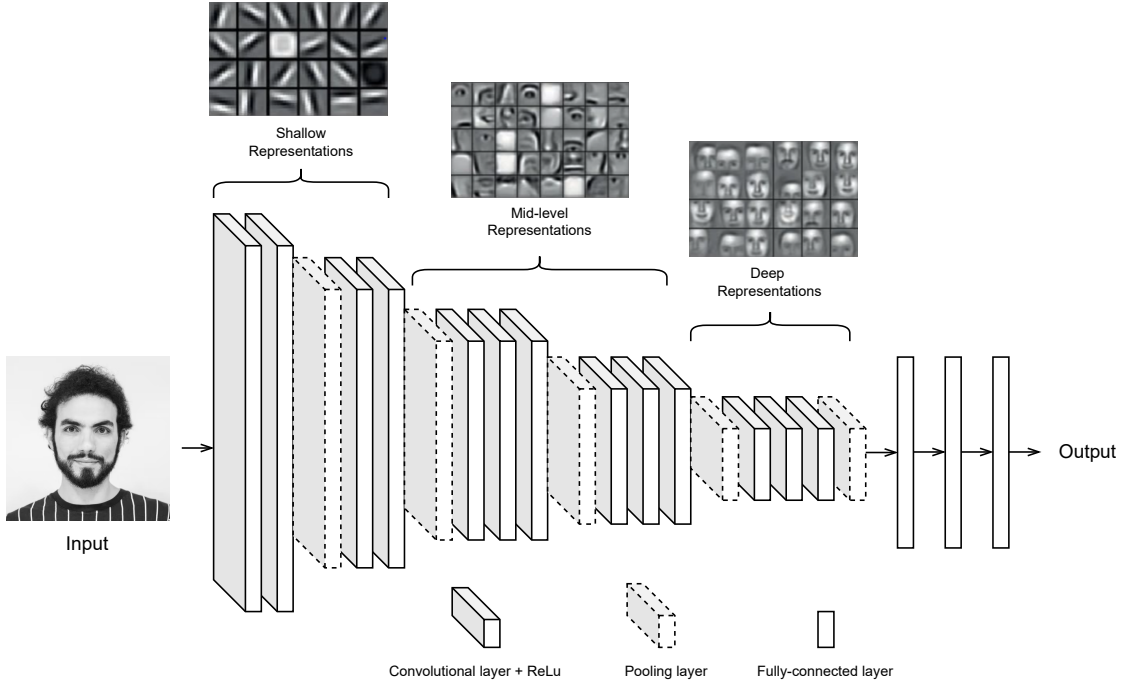


Figure 6.1: An illustration of the multi-granular nature of representations that are learned across shallow to deep layers in a network. The network architecture shown is of VGG16 [34].

source pre-trained model to the target dataset. As a direct extension of prior work, we propose the \mathcal{N} LEEP+ score where we extract representations from all layers of a source pre-trained model through a single, forward-pass of the target dataset input samples. The extracted representations are passed through PCA to compress the high-dimensional representation vectors for a compact representation and a common vector length across all representations. The PCA compressed representations are used to train a GMM clustering model. The result of the GMM model is the posterior probability, $\hat{P}_l(v|x)$, where l reflects the network layer from which the GMM posterior probability was estimated. The $\hat{P}_l(v|x)$ is used in Eq. 6.3 to compute the \mathcal{N} LEEP score of layer l . Finally, we average the \mathcal{N} LEEP scores for layers $l = \{1, \dots, L\}$ to return the proposed \mathcal{N} LEEP+ score, shown in Eq. 6.4.

$$\mathcal{N}\text{LEEP}+(\theta, \mathcal{D}) = \frac{1}{L} \sum_{l=1}^L \left(\frac{1}{n} \sum_{i=1}^n \log \left(\sum_{z \in \mathcal{Z}} \hat{P}(y_i|z)' \hat{P}_l(v|x) \right) \right) \quad (6.4)$$

6.3.3 Interpretable Attention Networks for Learning Representation Importance

Intuition dictates that different representation granularities will define different relation elements between source pre-trained models and a target dataset. In the previous section, we proposed an improved transferability measure, \mathcal{N} LEEP+, that evaluated the extreme case of using all layer representations from a pre-trained

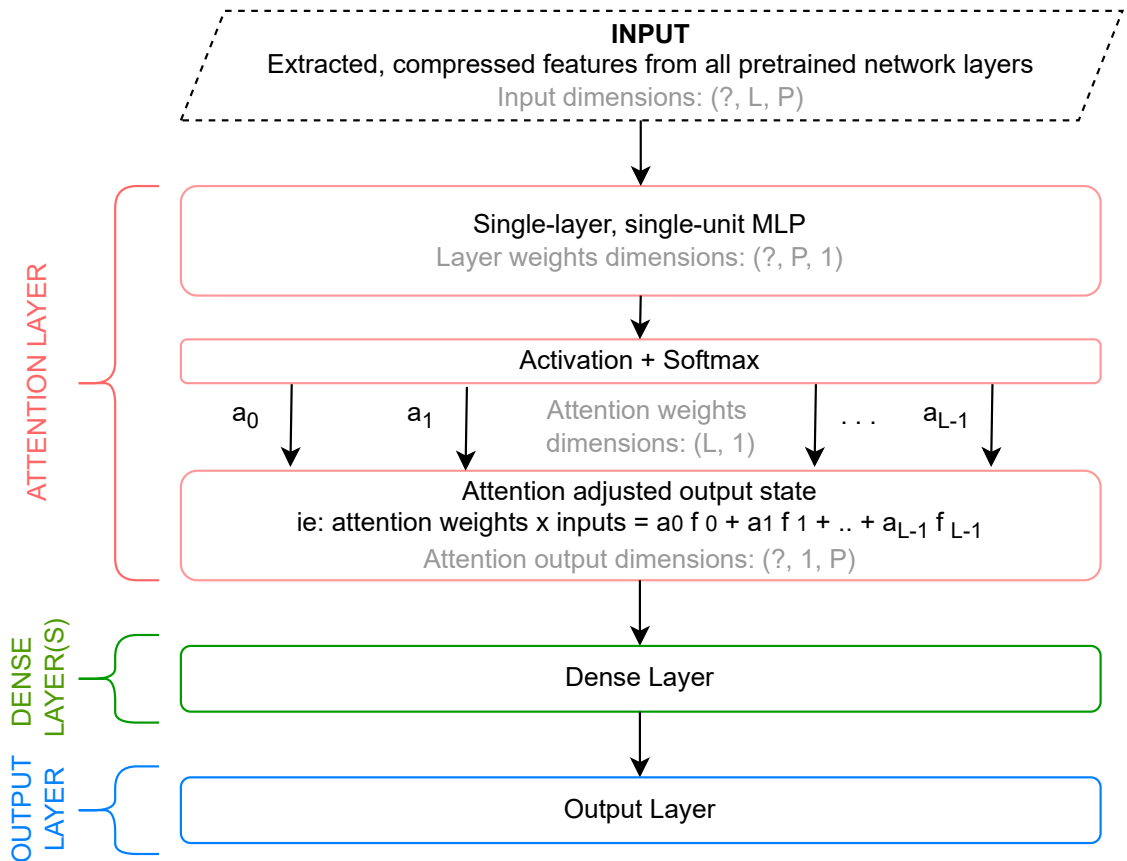


Figure 6.2: Attention network architecture for interpreting the importance of source pre-trained model representations. L is the number of layers in the source pre-trained network, P is the size of the low-dimensional input representation after being compressed through PCA, and '?' reflects an arbitrary input batch size.

model to map the transferability relationship between a source task and a target task. However, with the large parameter size of pre-trained models in literature today, it becomes costly to extract representations and learn a GMM model across all layers that could run high in count and dimension. Thus, in this section, we investigate the following question: ***Which layer representations will contribute the most to the transferability estimation between a source pre-trained model and a target dataset?***

Let's take two example scenarios of how a source and target task transferability relationship may vary. In the first example, we are given a set of source and target tasks that are derived from a similar input domain and a similar output domain, such as having both tasks' datasets composed of camera-captured images with class labels of the object in each image. In another example, we could be dealing with source and target tasks that belong to a similar output domain but a different input domain. For example, the source task may contain input image samples that are captured by a camera, while the target task input images are hand-drawn images of objects to be classified. The output domain is similar whereas we experience an

input distribution shift between the source and target. The assumption we make is that the two scenarios will reflect a different transferability relationship given the distribution shift that exists across the two distinct relationships.

Thus, we propose to learn an attention network that maps the PCA compressed target representations from all layers of the source pre-trained model to the target dataset output labels. The loss function of the attention network is being optimized with the cross-entropy loss function to correctly classify the true target dataset outputs. The attention network will learn a set of weights that combine linearly to produce a weighted output from the attention layer to map correctly to the output target labels. The attention weights that are learned are directly interpretable and can be used to map the importance of the layer representation to the final target output. Figure 6.2 showcases the attention network architecture of our proposed work, where L is the number of layers in the source pre-trained network and P is the size of the low-dimensional input representation after being compressed through PCA.

We build upon the \mathcal{N} LEEP+ measure by proposing *interpretable* LEEP (or iLEEP) where we do not use all layer representations for an improved scoring of transferability, but instead, we use the layer representations that are assigned an attention weight larger than the average attention weighting across all layers. As a result, we present a measure of optimized compromise between the computational complexity (where we need to compute a forward pass, run PCA compression, and train a GMM model for all layer representations) and the improved performance of looking beyond only the penultimate layer of a source pre-trained network, which is limited in representing the transferability estimation across varying source-target distribution shifts. The iLEEP measure is presented in Eq. 6.5.

$$\text{iLEEP}(\theta, \mathcal{D}) = \frac{1}{\sum_S 1} \sum_{l=1}^L a_l \left(\frac{1}{n} \sum_{i=1}^n \log \left(\sum_{z \in \mathcal{Z}} \hat{P}(y_i|z)' \hat{P}_l(v|x) \right) \right) \quad (6.5)$$

$$a_l = \begin{cases} a_l & \text{if } a_l > \frac{1}{l} \sum_{l=1}^L a_l \\ 0 & \text{otherwise.} \end{cases}$$

$$S = \{l : a_l \neq 0\}$$

Both the \mathcal{N} LEEP+ and iLEEP measures adopt the characteristics of LEEP, where they are upper bounded by zero and negative. The larger the measure score, that is, the smaller the absolute value of the score, the better the transferability indicated between the given source pre-trained model and target dataset.

6.4 Data & Experimental Setup

We evaluated our proposed transferability measures under the supervised task of object recognition using a pre-trained model and dataset benchmarks that are heavily utilized and evaluated in the literature. As such, we can guarantee the effectiveness

of our proposed methods on well-established benchmarks, compare them to the prior state-of-the-art, and set a solid ground for future explorations.

We use the Pearson correlation coefficient to evaluate the performance of a given transferability measure between a source pre-trained model and a target data. The Pearson correlation coefficient is computed between the transferability measure and the transfer accuracy, which is the final test accuracy on the source pre-trained model after it has been fine-tuned for the given target task.

6.4.1 *Implementation Details*

The proposed methods were implemented in Python using Google’s Keras Tensorflow deep learning framework. For experimental evaluation, we assess the performance of our transferability measures by computing the Pearson correlation and Kendall tau correlation [89] between the resulting \mathcal{N} LLEP+ and iLLEP measures for a source pre-trained model and target dataset pair and the final transfer accuracy of the pre-trained model after it is optimized for the target dataset. For transfer accuracy, we consider two settings: retraining the pre-trained model head and fine-tuning the entire pre-trained model layers with the target dataset.

6.4.2 *Datasets*

CIFAR10 [26] is an object recognition dataset composed of real images and their accompanied class label for the object contained in the image. CIFAR10 is a subset of the Imagenet dataset and contains 50,000 training samples and 10,000 testing samples of (32x32x3) dimension. The dataset is balanced and is composed of 10 class labels. All experimental results on CIFAR10 are the result of five averaged trial runs.

DomainNet [27] is an object recognition dataset composed of images from 6 different input domains including clipart, quickdraw, sketch, painting, infographic, and real (or camera-captured) images. The sample count per domain is 48,129, 172,500, 69,128, 72,266, 51,605, and 172,947, respectively. The dataset contains images of (300x300x3) dimension, is imbalanced, and is composed of 345 class labels per domain. We evaluate our measures under 4 domains of DomainNet: clipart, quickdraw, sketch, and painting. Since our source model is pre-trained on ImageNet, we eliminate the real domain since we use DomainNet to evaluate domains that differ from real, camera-captured images. We also eliminate the infographic domain due to high noise in labels. All experimental results on DomainNet are the result of five averaged trial runs.

6.4.3 *Pre-trained Models*

In our experiments, we consider two pre-trained model architectures: VGG16 [34] and ResNet18 [90]. Both models have been pre-trained on ImageNet [33], which is composed of more than a million samples of real, camera-captured images with 1000 class labels.

6.4.4 *Transfer Learning Algorithms*

When evaluating the transferability measures, we evaluate against the final transfer accuracy of the source pre-trained model on the target task for two transfer learning algorithms. First, we consider a transfer learning algorithm where we only retrain the head of the pre-trained model from scratch on the target task, which we refer to as *Retrain Head* in the table results. Second, we consider the transfer learning algorithm where we fine-tune the entire source pre-trained model weights with the target task, which we refer to as *Fine-tune* in the table results.

6.4.5 *Source-Target Settings*

We present experimental results under two source-target settings. The first we refer to as *standard setting*, describing the scenario where both the source and target tasks share similar input domain and output label distributions. Under this setting, we consider ImageNet and CIFAR10 as the source and target tasks, respectively, where they both share a similar input domain of camera-captured, real images and a similar label distribution of object classes. This is specifically the case since CIFAR10 is a subset of the ImageNet dataset. The second setting is the *cross-domain setting* where the source and target tasks share a similar output label domain but have a distribution shift in the input domain. Under this setting, we consider the scenario where our source task is ImageNet and the target task is DomainNet. While both datasets share a similar output label distribution of object classes, DomainNet contains images from different input domains such as sketched or painted images.

6.4.6 *Baselines*

We compared the proposed measures against the state-of-the-art LEEP score [17] and \mathcal{N} LEEP [25]. By outperforming LEEP, we guarantee improvements against more basic measures of transferability such as NCE [91] and H scores [92], which were shown to be outperformed by our LEEP baseline through extensive experiments in the prior work of Nguyen et al. [17].

6.5 Results & Discussion

6.5.1 *Transferability Measures vs. Transfer Accuracy*

Table 6.1 presents the results of Pearson correlation coefficient scores for the proposed and state-of-the-art transferability measures. Our results show that our proposed iLEEP and \mathcal{N} LEEP+ are consistently effective in measuring the transferability between VGG16 and ResNet18 pre-trained models and the target CIFAR10 dataset, under the standard setting.

Evaluation against State-of-the-Art

We can see from Table 6.1 that both our proposed measures consistently outperform state-of-the-art, LEEP and \mathcal{N} LEEP. We witness consistent improvement

Table 6.1: Comparison of the Pearson correlation coefficients of iLEEP, \mathcal{N} LEEP+, \mathcal{N} LEEP, and LEEP on the standard setting of the source and target data having similar domain and label distributions.

TL Algorithm	Source model	Source data	Target data	Target properties	LEEP [17]	\mathcal{N} LEEP [25]	\mathcal{N} LEEP+	iLEEP	iLEEP vs \mathcal{N} LEEP+ Δ_{coef}
Retrain Head	VGG16	ImageNet	CIFAR10	Similar labels, similar domains, balanced.	0.878	0.903	0.984	0.973	-1.11%
	ResNet18				0.822	0.897	0.956	0.944	-1.26%
Fine-tune	VGG16	ImageNet	CIFAR10	Similar labels, similar domains, balanced, small.	0.841	0.899	0.988	0.981	-0.71%
	ResNet18				0.825	0.870	0.983	0.979	-0.41%
Retrain Head	VGG16	ImageNet	CIFAR10	Similar labels, similar domains, balanced, small.	0.654	0.831	0.930	0.895	-3.76%
	ResNet18				0.589	0.844	0.926	0.872	-5.83%
Fine-tune	VGG16	ImageNet	CIFAR10	Similar labels, similar domains, balanced, small.	0.562*	0.796	0.877	0.849	-3.19%
	ResNet18				0.641*	0.802	0.858	0.822	-4.19%

* Not statistically significant with p-value > 0.05.

in the Pearson correlation coefficients of iLEEP in comparison to both prior state-of-the-art measures. More so, \mathcal{N} LEEP+ shows the best performance in terms of the Pearson correlation values. This, however, comes at the expense of added complexity in the extraction of network representations across all layers, as opposed to LEEP and \mathcal{N} LEEP, which only require extracting representations from only the output classification or penultimate layer, respectively.

iLEEP vs. \mathcal{N} LEEP+ The last column of Table 6.1 highlights the percentage decrease in the Pearson correlation coefficient of iLEEP in comparison to \mathcal{N} LEEP+. Under the standard setting evaluated on CIFAR10, we can see that the percentage decrease of iLEEP is no more than -1.26% across 4 experimental setups. Under the standard setting with a small data sample size where $N/K=10$ (N is the number of training samples and K is the number of unique class labels), the percentage decrease of iLEEP does not exceed -5.83% in comparison to \mathcal{N} LEEP+ across 4 experimental setups.

The aforementioned percentage decrease is a result of a -52.2% and -47% decrease in the dimension of representations extracted to compute iLEEP in comparison to \mathcal{N} LEEP+. As a result, iLEEP requires less storage and computational resources. This highlights the promising performance of iLEEP as it presents a compromise between the enhanced transferability capabilities of \mathcal{N} LEEP+ and the computational advantage of LEEP and \mathcal{N} LEEP.

6.5.2 Evaluation under the Cross-domain Setting

Table 6.2 presents the results of Pearson correlation coefficient scores for the proposed and state-of-the-art transferability measures under the cross-domain setting. The presented results highlight the strength of the proposed iLEEP and \mathcal{N} LEEP+ measures in evaluating transferability where a distribution shift exists across the input domain of the source pre-trained model and the target dataset. Specifically, this strength is highlighted in comparison to the performance of both previous LEEP \mathcal{N} LEEP measures, which display a much lower correlation between their transferability estimation and the transfer accuracy under the cross-domain setting. This behavior is consistent across 4 domains of the DomainNet dataset. As a result, we conclude that through extracting richer representations of multi-granular features

Table 6.2: Comparison of the Pearson correlation coefficients of iLEEP, \mathcal{N} LEEP+, \mathcal{N} LEEP, and LEEP on the cross-domain setting where the source and target data belong to different input domain distributions.

TL Algorithm	Source model	Source data	Target data	Target properties	LEEP [17]	\mathcal{N} LEEP [25]	\mathcal{N} LEEP+	iLEEP
Retrain Head	VGG16	ImageNet	DomainNet/Clipart	Cross-domain, imbalanced.	0.668	0.721	0.879	0.829
	ResNet18				0.588	0.721	0.911	0.809
Fine-tune	VGG16	ImageNet	DomainNet/Clipart	Cross-domain, imbalanced.	0.605	0.723	0.930	0.880
	ResNet18				0.513	0.750	0.920	0.912
Retrain Head	VGG16	ImageNet	DomainNet/Quickdraw	Cross-domain, imbalanced.	0.566	0.673	0.819	0.754
	ResNet18				0.605*	0.646	0.889	0.784
Fine-tune	VGG16	ImageNet	DomainNet/Quickdraw	Cross-domain, imbalanced.	0.564	0.644	0.826	0.809
	ResNet18				0.557*	0.649	0.859	0.795
Retrain Head	VGG16	ImageNet	DomainNet/Sketch	Cross-domain, imbalanced.	0.565	0.780	0.884	0.827
	ResNet18				0.618	0.682	0.887	0.833
Fine-tune	VGG16	ImageNet	DomainNet/Sketch	Cross-domain, imbalanced.	0.572	0.688	0.897*	0.873
	ResNet18				0.583	0.673	0.870	0.861
Retrain Head	VGG16	ImageNet	DomainNet/Painting	Cross-domain, imbalanced.	0.532*	0.741	0.926	0.814
	ResNet18				0.626	0.736	0.820	0.780
Fine-tune	VGG16	ImageNet	DomainNet/Painting	Cross-domain, imbalanced.	0.566	0.696	0.840	0.776
	ResNet18				0.536	0.693	0.866	0.823

* Not statistically significant with p-value > 0.05.

from the source pre-trained model, our proposed iLEEP and \mathcal{N} LEEP+ measures are capable of handling cross-domain settings in comparison to previous work.

6.5.3 Evaluation against Varying Design Parameters

Impact of Architecture

We run experiment trials under two different source pre-trained models: VGG16 and ResNet18. The results are shown in Table 6.1 and 6.2. By analyzing the results, it is clear that the effectiveness of the proposed transferability measures is consistent across varying architectures. On average, the Pearson correlation coefficients of the ResNet18 model are slightly lower than those of VGG16. However, the presented experiments cannot present conclusive findings on how the depth of the pre-trained model architecture may impact performance. Nevertheless, both iLEEP and \mathcal{N} LEEP+ are consistently capable of measuring the success of transfer from the source pre-trained models to the target task and are consistently outperforming previous state-of-the-art under both architectures.

Impact of Sample Size

The second set of experimental trials in Table 6.1 is conducted on a small data regime ($N/K=10$). We again see consistent improvement in the proposed iLEEP and \mathcal{N} LEEP+ measures in comparison to previous work, however, we detect a higher percentage decrease in iLEEP compared to \mathcal{N} LEEP+ under the small data regime. This may be due to the limited information being captured across a smaller target sample size in iLEEP versus \mathcal{N} LEEP+, since \mathcal{N} LEEP+ is capturing a richer representation across all layers of the source pre-trained model. However, the percentage decrease remains significantly low (no lower than -5.83% in comparison to a 47%&-52.2% reduction in the dimensions of extracted representations between iLEEP and \mathcal{N} LEEP+).

Impact of Label Size

Table 6.1 presents results of the transferability measures on CIFAR10, which contains 10 class labels, whereas Table 6.2 presents results on DomainNet, which contains 345 class labels per domain. For the two experiment setups, we see consistent effectiveness of the proposed measures in measuring transferability and in outperforming previous state-of-the-art across the varying size of the class label sets of CIFAR10 and DomainNet.

6.5.4 Interpretation of *iLEEP*

As we investigated the attention-based representations learned through the *iLEEP* measure, we found some interesting behavior that was consistent across the experimental trials run under the standard setting and the cross-domain setting.

Under the standard setting with CIFAR10 as the target dataset, the attention scores learned by the attention-based network consistently highlighted the relevance of the first 1-2 and the last 3-4 convolutional layers with both VGG16 and ResNet18 architectures. Intuitively, the focus on the last convolutional blocks and layers is expected as the high-level representations are likely to be shared in a source-target transfer where the label distributions are similar. It was interesting to also see shallow layers highlighted under this setting, reflecting that there are indeed shared low-level features being represented across both source and target domains. This, again, aligns with intuition since CIFAR10 is a subset of the source ImageNet dataset on which the pre-trained models were trained. Thus, the input distributions are shared, and thus, relevant in assessing transferability.

Moreover, the cross-domain setting with DomainNet also witnessed similar consistent behavior in the attention scores assigned to the representations extracted for evaluating transferability. Under the cross-domain setting, the first 1-4 and the last 3-4 convolutional layers with both VGG16 and ResNet18 architectures were scored as relevant to measuring transferability. The focus on the last convolutional blocks and layers is aligned with the intuition from the standard setting since we are still dealing with similar label distributions across DomainNet and ImageNet. However, it was interesting to see that high relevance was given to more shallow layers. This highlights that low-level features seem to serve as strong identifiers of how close the input distributions of the source and target are, and as a result, strong identifiers of whether transferability across source and target will be high or low.

6.5.5 Applications of *iLEEP* and *NLEEP+*

Our proposed *iLEEP* and *NLEEP+* measures quantify the success of transferability of knowledge between a given source pre-trained model and a target dataset. As a result, a direct consequence application is to use the measures in the selection of the best model for a target dataset from a zoo of given source pre-trained models.

Aside from pre-trained model selection, the transferability measures can prove to be useful in continual learning [83] where a sequence of incoming tasks are to be learned and choosing the order in which to introduce tasks is important. Moreover,

the measures may also be extended to support multitask learning scenarios [38] as well where multiple target tasks are learned in parallel.

6.6 Broader Applications

The proposed metrics in this chapter were evaluated on a computer vision benchmark of object recognition as a standard benchmark for the state-of-the-art. With a particular focus on time-series applications under this work, the next extension required to evaluate these metrics on time-series applications is to develop rich, benchmark time-series pre-trained models, which are lacking in literature today. Moreover, it is also important to extend the empirical study to evaluate source-target relationships where a label shift exists (e.g., using ImageNet object recognition as the source pre-trained model and ImageNet for object localization as the target dataset) and where a cross-domain, cross-label shift exists (e.g., using ImageNet object recognition as the source pre-trained model and PASCAL VOC [93] for semantic segmentation as the target dataset).

The applicability of the metrics to new pre-trained models and datasets is straightforward since the metric computation relies on extracting the network representations from different layers in the architecture. Thus, there is no restriction on the type of network, including fully-connected, convolutional, recurrent, or transformer architectures. Consequently, the developed metrics can also be extended to evaluating the transferability of state-of-the-art language models, which are heavily utilized in research and production.

CHAPTER 7

CONCLUSION & FUTURE DIRECTIONS

In summary, this dissertation aims to use existing resources, such as pre-trained ML models, to improve the performance when learning new tasks that have limited labeled time-series data. Many scenarios arise where pre-trained models need to be fine-tuned to adapt to new tasks or information. Transfer learning presents a promising paradigm of techniques to improve performance on new target tasks, nevertheless, there remain several open challenges to achieving optimal transfer learning. We focus on three research challenges that address common scenarios faced in transfer learning settings.

7.1 Multitask Learning Models for Time-series Data

7.1.1 *Summary*

We addressed the labeled data deficiency bottleneck in building personalized models for a group of target tasks. We presented a new systematic approach for designing, evaluating, and improving Multitask Learning (MTL) models for time-series data. These MTL models learn multiple target tasks simultaneously and leverage information transfer across all tasks. We consider three primary design components: features capturing the time dynamics in data, similarity metrics reflecting degrees of commonality and uniqueness across entities, and generalization metrics to prevent overfitting. The proposed framework enables the introduction of efficient new MTL models and advances the prior state-of-the-art. We successfully applied and tested the design framework, and as a result, presented a MTL deep learning model that makes use of a hierarchical architecture of Convolutional Neural Network (CNN) and Gated Recurrent Unit (GRU) layers. The resulting MTL deep learning model that was designed using the proposed framework was evaluated on 4 benchmark human activity recognition datasets and was shown to outperform a collection of previous state-of-the-art methods.

7.1.2 *Open Research Directions*

Two prominent challenges remain open in building MTL models. The first challenge is related to the computational complexity associated with adding more tasks in

MTL models. We explore this research direction in the second objective of the work by investigating how we can minimize catastrophic forgetting when fine-tuning a pre-trained model with a single new task. However, it is possible to approach this question from a different approach, such as looking into compressing the number of tasks to train by clustering tasks into related groups. Hierarchical clustering can be used to break down the computational complexity of training a large number of tasks in parallel in an MTL architecture, for instance. With such a direction of work, several questions may arise such as (1) which tasks should be clustered and learned together? (2) should tasks be clustered by looking at the raw, data level or should intermediate representations be extracted to better represent the relatedness of tasks? (3) what type of clustering relationships can be used to define relatedness across tasks?

The second challenge is related to establishing a formal characterization of the relationship across multiple target tasks trained in the MTL models. In this dissertation, we study a transferability estimation metric under objective three which looks at the relationship between a source pre-trained model and a single target task. However, with multiple target tasks introduced to a model, the transferability relationship becomes more complex, and it is vital to understand how the target tasks impact one another, positively and negatively, that is boosting or deteriorating the performance of other tasks.

7.2 Catastrophic Forgetting in Transfer Learning

7.2.1 *Summary*

We considered the scenario where we seek to adapt pre-trained models that had been previously trained on a source task, where the data is no longer accessible, to new target tasks that have limited labeled data. We wish to improve performance on the target task while maintaining performance on the source task. The challenge is to overcome the effect of catastrophic forgetting in the pre-trained neural network while fine-tuning the network on the new target task data. We proposed LOMA, a Lifelong Multitask Autoencoder network, that is trained using a multi-objective loss function with knowledge distillation, prediction error, and reconstruction error losses. We evaluated the proposed approach on sensing data under human activity recognition tasks where evaluations are run under four benchmark datasets. Experimental results showed minimized catastrophic forgetting compared to baseline methods of fine-tuning and multitask learning as well as prior state-of-the-art LwF. An ablation analysis study highlighted the relevance of the proposed knowledge distillation and reconstruction losses by showing an average improvement of +3.5% on preserving old tasks. Moreover, when we evaluated the performance of learning new tasks in sequence, we found that the order in which new tasks are introduced sequentially matters.

7.2.2 *Open Research Directions*

A direct extension of the work conducted under this research objective is to investigate methods that would learn the optimal order in which tasks are introduced to a model under a continual learning setting, where multiple tasks are queued to be learned. Moreover, there remains open space to explore the best combination and weighting of the multiple loss functions of the defined multi-objective learning function in Eq. 5.1 of Chapter 5.

Another open challenge that arises from the first objective of the work is the computational complexity of introducing a large number of tasks, which is consequently represented by additional output layer parameters per task. Once again, future work can investigate introducing clusters of new tasks as opposed to a single task at a time. Thus, a new output parameter branch would be learned for the entire cluster of new tasks, as opposed to introducing a new set of parameters per unique task.

7.3 Measures of Transferability Estimation

7.3.1 *Summary*

We studied the transferability estimation problem between a pre-trained model trained on a source task and a new target dataset. Here, the goal in such a scenario is to determine a priori and without the need for expensive optimization if the source pre-trained model will transfer knowledge successfully to the new target task and, as a result, render an accurate model fine-tuned for the target task. We propose a new method to learn which feature representations extracted from a source pre-trained model are most descriptive of the source and target task transferability relationship, thus, eliminating the restriction on relying on final model outputs only. We present an interpretable transferability measure, iLEEP, that uses an attention-based network to learn the optimal combination of pre-trained model representations that hold the highest contribution to transferability across the source and target tasks. The proposed work was evaluated on two benchmark computer vision datasets of object recognition, and the experimental results showed that our proposed iLEEP measure has superior performance to two prior state-of-the-art analytical measures, LEEP and \mathcal{N} LEEP, with the added benefit of being intuitive and interpretable.

7.3.2 *Open Research Directions*

In order to extend the transferability measures to time-series-based models, a need arises for developing benchmark, rich pre-trained time-series models. There is a lack of such models in literature today. It has become essential to create such standards for the sensing domain. It is also important to establish a clear understanding of the type of pre-trained models needed for time-series. It is also important to identify the application of time-series (if there is a specific one) that could serve as a ground foundation for all others, that is, similar to what we see with object recognition for

the computer vision domain and with text generation/classification for the natural language processing domain.

Moreover, future directions can investigate the standardization of the interpretation derived from iLEEP in the layers that contribute to the transferability estimation between the pre-trained source model and the target dataset. That is, we need to understand if there are specific layers that define the relation of transferability under various distribution shift settings of cross-domain, cross-task, common domain and task, and cross-domain and cross-task relations. Finally, future work can look into the extension of iLEEP and its feasibility for new classes of tasks such as unsupervised and semi-supervised tasks.

APPENDIX A

DATA STATISTICS OF CHAPTER 5 EXPERIMENTS

Table [A.1](#) presents summary of statistics on datasets used in our experiments of Chapter 5 specifying the count statistics, splits between training and testing, category of labels, and distribution of labels across the dataset.

At each experimental setup, one dataset is selected to serve as the old tasks' domain and another is selected to serve as the new tasks' domain. When dividing the dataset into training and testing, user subjects (or tasks) are equally sampled into each subset along with the different label categories through the use of stratified sampling techniques.

Table A.1: Summary of data statistics for ALKAN, DAS, OPPORTUNITY, and UCI HAR.

Dataset	Count Statistics	Data Splits	Label Category	Distribution of Labels <i>[Activity ID]:[Perc. of Samples]:[Description]</i>
ALKAN [12]	Total Sample Count: 167,391 Total Tasks: 20 Labels Count: 7	Train: 90% Test: 10%	Locomotion activity	A1: 13: Walk A2: 16: Stand A3: 19: Lie A4: 21: Sit A5: 9: Walking upstairs A6: 9: Walking downstairs A7: 13: Run
DAS [18]	Total Sample Count: 35,625 Total Tasks: 8 Labels Count: 19	Train: 80% Test: 20%	Locomotion and high-level activity	A1: 7: Sitting A2: 8: Standing A3: 8: Lying on back A4: 9: Lying on right side A5: 5: Ascending stairs A6: 5: Descending stairs A7: 4: Standing in an elevator still A8: 3: Moving around in an elevator A9: 5: Walking in a parking lot A10: 7: Walking on a treadmill in flat position A11: 4: Walking on a treadmill in inclined position A12: 5: Running on a treadmill with a speed of 8 km/h A13: 5: Exercising on a stepper A14: 6: Exercising on a cross trainer A15: 7: Cycling on an exercise bike in horizontal position A16: 5: Cycling on an exercise bike in vertical position A17: 2: Rowing A18: 3: Jumping A19: 2: Playing basketball
OPP [19]	Total Sample Count: 35,312 Total Tasks: 4 Labels Count: 9	Train: 80% Test: 20%	High-level activity	A1: 14: Open and close the fridge A2: 13: Open and close the dishwasher A3: 11: Open and close 3 drawers A4: 16: Open and close door 1 A5: 14: Open and close door 2 A6: 10: Turn on and off the lights A7: 9: Clean table A8: 6: Drink (standing) A9: 7: Drink (sitting)
OPP [19]	Total Sample Count: 15,810 Total Tasks: 4 Labels Count: 4	Train: 80% Test: 20%	Locomotion activity	A1: 24: Walk A2: 19: Run A3: 27: Lie A4: 30: Sit
UCI [20]	Total Sample Count: 20,598 Total Tasks: 30 Labels Count: 6	Train: 70% Valid: 10% Test: 20% <i>*This dataset was used for the model's hyperparameter tuning.</i>	Locomotion activity	A1: 19: Walk A2: 14: Stand A3: 15: Lie A4: 23: Sit A5: 14: Walking upstairs A6: 15: Walking downstairs

BIBLIOGRAPHY

- [1] K. Jordan, J. H. Challis, and K. M. Newell, “Walking speed influences on gait cycle variability,” *Gait & posture*, vol. 26, no. 1, pp. 128–134, 2007.
- [2] R. Cowie, E. Douglas-Cowie, N. Tsapatsoulis, *et al.*, “Emotion recognition in human-computer interaction,” *IEEE Signal processing magazine*, vol. 18, no. 1, pp. 32–80, 2001.
- [3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [4] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, “Cell phone-based biometric identification,” in *2010 Fourth IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, IEEE, 2010, pp. 1–7.
- [5] H.-Y. Chang, Z. Li, S. Das, *et al.*, “A personalized pacing system for real-time physical activity advising,” in *2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, IEEE, 2017, pp. 266–267.
- [6] Rich Caruana, “Multitask learning: A knowledge-based source of inductive bias,” *Proc. of the 10th Int’l Conference in Machine Learning*, 266–267, Jul. 2017.
- [7] X. Sun, H. Kashima, and N. Ueda, “Large-scale personalized human activity recognition using online multitask learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2551–2563, 2012.
- [8] Liu, Pengfei and Qiu, Xipeng and Huang, Xuanjing, “Recurrent neural network for text classification with multi-task learning,” *arXiv preprint arXiv:1605.05101*, 2016.
- [9] R. Chavarriaga, H. Sagha, A. Calatroni, *et al.*, “The opportunity challenge: A benchmark database for on-body sensor-based activity recognition,” *Pattern Recognition Letters*, vol. 34, no. 15, pp. 2033–2042, 2013.
- [10] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “A public domain dataset for human activity recognition using smartphones.” in *Esann*, 2013.

- [11] B. Barshan and M. C. Yükses, “Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units,” *The Computer Journal*, vol. 57, no. 11, pp. 1649–1667, 2014.
- [12] Y. Hattori, S. Inoue, and G. Hirakawa, “A large scale gathering system for activity data with mobile sensors,” in *2011 15th annual international symposium on wearable computers*, IEEE, 2011, pp. 97–100.
- [13] M. De Lange, R. Aljundi, M. Masana, *et al.*, “A continual learning survey: Defying forgetting in classification tasks,” *arXiv preprint arXiv:1909.08383*, 2019.
- [14] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, *et al.*, “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.
- [15] Z. Li and D. Hoiem, “Learning without forgetting,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, 2018.
- [16] A. Rannen, R. Aljundi, M. B. Blaschko, and T. Tuytelaars, “Encoder based lifelong learning,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1320–1328.
- [17] C. Nguyen, T. Hassner, M. Seeger, and C. Archambeau, “Leep: A new measure to evaluate transferability of learned representations,” in *International Conference on Machine Learning*, PMLR, 2020, pp. 7294–7305.
- [18] B. Barshan and M. C. Yükses, “Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units,” *The Computer Journal*, vol. 57, no. 11, pp. 1649–1667, 2014.
- [19] R. Chavarriaga, H. Sagha, A. Calatroni, *et al.*, “The opportunity challenge: A benchmark database for on-body sensor-based activity recognition,” *Pattern Recognition Letters*, vol. 34, no. 15, pp. 2033–2042, 2013.
- [20] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “A public domain dataset for human activity recognition using smartphones,” in *Esann*, vol. 3, 2013, p. 3.
- [21] T. Standley, A. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese, “Which tasks should be learned together in multi-task learning?” In *International Conference on Machine Learning*, PMLR, 2020, pp. 9120–9132.
- [22] B. Neyshabur, H. Sedghi, and C. Zhang, “What is being transferred in transfer learning?” *Advances in neural information processing systems*, vol. 33, pp. 512–523, 2020.
- [23] X. Su, Y. Jiang, S. Guo, and F. Chen, “Task understanding from confusing multi-task data,” in *International Conference on Machine Learning*, PMLR, 2020, pp. 9177–9186.

- [24] K. You, Y. Liu, J. Wang, and M. Long, “Logme: Practical assessment of pre-trained models for transfer learning,” in *International Conference on Machine Learning*, PMLR, 2021, pp. 12 133–12 143.
- [25] Y. Li, X. Jia, R. Sang, *et al.*, “Ranking neural checkpoints,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2663–2673.
- [26] A. Krizhevsky, G. Hinton, *et al.*, “Learning multiple layers of features from tiny images,” 2009.
- [27] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, “Moment matching for multi-source domain adaptation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1406–1415.
- [28] T. M. Mitchell and T. M. Mitchell, *Machine learning*, 9. McGraw-hill New York, 1997, vol. 1.
- [29] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, 4. Springer, 2006, vol. 4.
- [30] S. Saha, *A comprehensive guide to convolutional neural networks — the eli5 way*, Available at <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> (2018/12/15).
- [31] S. Kostadinov, *Understanding gru networks*, Available at <https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be> (2017/12/16).
- [32] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.
- [33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.
- [34] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [35] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, “Transfer learning for time series classification,” in *2018 IEEE International Conference on Big Data (Big Data)*, IEEE, 2018, pp. 1367–1376.
- [36] R. Caruana, “Multitask learning,” *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.
- [37] Y. Zhang and D.-Y. Yeung, “A regularization approach to learning task relationships in multitask learning,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 8, no. 3, pp. 1–31, 2014.
- [38] R. A. Mahmoud, H. Hajj, and F. N. Karameh, “A systematic approach to multi-task learning from time-series data,” *Applied Soft Computing*, vol. 96, p. 106 586, 2020.

- [39] X. Lu, Z. Yu, C. Liu, Y. Liu, H. Xiong, and B. Guo, “Inferring lifetime status of point-of-interest: A multitask multiclass approach,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 14, no. 1, pp. 1–27, 2020.
- [40] G. M. Weiss and J. W. Lockhart, “Identifying user traits by mining smart phone accelerometer data,” in *Proceedings of the fifth international workshop on knowledge discovery from sensor data*, ACM, 2011, pp. 61–69.
- [41] R. Paradiso, A. Bianchi, K. Lau, and E. Scilingo, “Psyche: Personalised monitoring systems for care in mental health,” in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, IEEE, 2010, pp. 3602–3605.
- [42] M. Qiu, P. Zhao, K. Zhang, *et al.*, “A short-term rainfall prediction model using multi-task convolutional neural networks,” in *Data Mining (ICDM), 2017 IEEE International Conference on*, IEEE, 2017, pp. 395–404.
- [43] R.-G. Cirstea, D.-V. Micu, G.-M. Muresan, C. Guo, and B. Yang, “Correlated time series forecasting using multi-task deep neural networks,” *CIKM*, 2018.
- [44] K. Hara, K. Inoue, K. Takanashi, and T. Kawahara, “Prediction of turn-taking using multitask learning with prediction of backchannels and fillers,” *Listener*, vol. 162, p. 364, 2018.
- [45] Harutyunyan, Hrayr and Khachatryan, Hrant and Kale, David C and Galstyan, Aram, “Multitask Learning and Benchmarking with Clinical Time Series Data,” *arXiv preprint arXiv:1703.07771*, 2017.
- [46] Yang, Zhilin and Salakhutdinov, Ruslan and Cohen, William W, “Transfer learning for sequence tagging with hierarchical recurrent networks,” *arXiv preprint arXiv:1703.06345*, 2017.
- [47] Tang, Zhiyuan and Li, Lantian and Wang, Dong, *Multi-task recurrent model for speech and speaker recognition*. IEEE, 2016, 1–4.
- [48] M. M. Hassan, M. Z. Uddin, A. Mohamed, and A. Almogren, “A robust human activity recognition system using smartphone sensors and deep learning,” *Future Generation Computer Systems*, vol. 81, pp. 307–313, 2018.
- [49] S. A. Rokni, M. Nourollahi, and H. Ghasemzadeh, “Personalized human activity recognition using convolutional neural networks,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [50] A. Ignatov, “Real-time human activity recognition from accelerometer data using convolutional neural networks,” *Applied Soft Computing*, vol. 62, pp. 915–922, 2018.
- [51] A. Robins, “Catastrophic forgetting, rehearsal and pseudorehearsal,” *Connection Science*, vol. 7, no. 2, pp. 123–146, 1995.
- [52] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

- [53] H. Shin, J. K. Lee, J. Kim, and J. Kim, “Continual learning with deep generative replay,” in *Advances in Neural Information Processing Systems*, 2017, pp. 2990–2999.
- [54] J. Xu and Z. Zhu, “Reinforced continual learning,” in *Advances in Neural Information Processing Systems*, 2018, pp. 899–908.
- [55] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, *et al.*, “Progressive neural networks,” *arXiv preprint arXiv:1606.04671*, 2016.
- [56] C. Fernando, D. Banarse, C. Blundell, *et al.*, “Pathnet: Evolution channels gradient descent in super neural networks,” *arXiv preprint arXiv:1701.08734*, 2017.
- [57] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, “Overcoming catastrophic forgetting with hard attention to the task,” *arXiv preprint arXiv:1801.01423*, 2018.
- [58] B. Pfülb and A. Gepperth, “A comprehensive, application-oriented study of catastrophic forgetting in dnns,” *arXiv preprint arXiv:1905.08101*, 2019.
- [59] F. Zenke, B. Poole, and S. Ganguli, “Continual learning through synaptic intelligence,” *Proceedings of machine learning research*, vol. 70, p. 3987, 2017.
- [60] S. Farquhar and Y. Gal, “Towards robust evaluations of continual learning,” *arXiv preprint arXiv:1805.09733*, 2018.
- [61] Q. Wang, L. Zhan, P. Thompson, and J. Zhou, “Multimodal learning with incomplete modalities by knowledge distillation,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1828–1838.
- [62] B. Neyshabur, H. Sedghi, and C. Zhang, “What is being transferred in transfer learning?” *Advances in neural information processing systems*, vol. 33, pp. 512–523, 2020.
- [63] T. Standley, A. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese, “Which tasks should be learned together in multi-task learning?” In *International Conference on Machine Learning*, PMLR, 2020, pp. 9120–9132.
- [64] X. Su, Y. Jiang, S. Guo, and F. Chen, “Task understanding from confusing multi-task data,” in *International Conference on Machine Learning*, PMLR, 2020, pp. 9177–9186.
- [65] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, “Gradient surgery for multi-task learning,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 5824–5836, 2020.
- [66] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese, “Taskonomy: Disentangling task transfer learning,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3712–3722.
- [67] B. Lindemann, T. Müller, H. Vietz, N. Jazdi, and M. Weyrich, “A survey on long short-term memory networks for time series prediction,” *Procedia CIRP*, vol. 99, pp. 650–655, 2021.

- [68] J. Yang, M. N. Nguyen, P. P. San, X. Li, and S. Krishnaswamy, “Deep convolutional neural networks on multichannel time series for human activity recognition,” in *Ijcai*, Buenos Aires, Argentina, vol. 15, 2015, pp. 3995–4001.
- [69] C. A. Ronao and S.-B. Cho, “Human activity recognition with smartphone sensors using deep learning neural networks,” *Expert systems with applications*, vol. 59, pp. 235–244, 2016.
- [70] R. D. Chambers and N. C. Yoder, “Filternet: A many-to-many deep learning architecture for time series classification,” *Sensors*, vol. 20, no. 9, p. 2498, 2020.
- [71] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, “Deepsense: A unified deep learning framework for time-series mobile sensing data processing,” in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 351–360.
- [72] J. Wang, T. Sun, B. Liu, Y. Cao, and H. Zhu, “Clvsa: A convolutional lstm based variational sequence-to-sequence model with attention for predicting trends of financial markets,” *arXiv preprint arXiv:2104.04041*, 2021.
- [73] T. Patterson, N. Khan, S. McClean, *et al.*, “Sensor-based change detection for timely solicitation of user engagement,” *IEEE Transactions on Mobile Computing*, vol. 16, no. 10, pp. 2889–2900, 2017.
- [74] Rich Caruana, “Multitask Learning,” *Machine Learning*, vol. 28, no. 1, 41–75, 1997.
- [75] N. Y. Hammerla, S. Halloran, and T. Plötz, “Deep, convolutional, and recurrent models for human activity recognition using wearables,” *arXiv preprint arXiv:1604.08880*, 2016.
- [76] X. Fan, H. Zhang, C. Leung, and C. Miao, “Comparative study of machine learning algorithms for activity recognition with data sequence in home-like environment,” in *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, IEEE, 2016, pp. 168–173.
- [77] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.
- [78] W. Min, B. Mott, J. Rowe, and J. Lester, “Deep lstm-based goal recognition models for open-world digital games,” in *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
- [79] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [80] H. Sagha, S. T. Digumarti, J. d. R. Millán, *et al.*, “Benchmarking classification techniques using the opportunity human activity dataset,” in *2011 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, 2011, pp. 36–40.

- [81] R. Saeedi, K. Sasani, S. Norgaard, and A. H. Gebremedhin, “Personalized human activity recognition using wearables: A manifold learning-based knowledge transfer,” in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, 2018, pp. 1193–1196.
- [82] J. Cao, W. Li, C. Ma, and Z. Tao, “Optimizing multi-sensor deployment via ensemble pruning for wearable activity recognition,” *Information Fusion*, vol. 41, pp. 68–79, 2018.
- [83] R. A. Mahmoud and H. Hajj, “Multi-objective learning to overcome catastrophic forgetting in time-series applications,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2022.
- [84] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network. nips deep learning workshop,” *arXiv preprint arXiv:1503.02531*, 2014.
- [85] Z. Gao, D. Liu, K. Huang, and Y. Huang, “Context-aware human activity and smartphone position-mining with motion sensors,” *Remote Sensing*, vol. 11, no. 21, p. 2531, 2019.
- [86] A. Ng *et al.*, “Sparse autoencoder,” *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.
- [87] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, “Deep learning for sensor-based activity recognition: A survey,” *Pattern Recognition Letters*, vol. 119, pp. 3–11, 2019.
- [88] X. Wang and J. G. Schneider, “Generalization bounds for transfer learning under model shift.,” in *UAI*, 2015, pp. 922–931.
- [89] M. G. Kendall, “A new measure of rank correlation,” *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.
- [90] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [91] A. T. Tran, C. V. Nguyen, and T. Hassner, “Transferability and hardness of supervised classification tasks,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1395–1405.
- [92] Y. Bao, Y. Li, S.-L. Huang, *et al.*, “An information-theoretic approach to transferability in task transfer learning,” in *2019 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2019, pp. 2309–2313.
- [93] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.