Foundations of Radio Frequency Transfer Learning

Lauren J. Wong

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Electrical Engineering

Alan J. Michaels, Chair
Harpreet S. Dhillon
Laura J. Freeman
Jia-Bin Huang
Ryan K. Williams

November 10, 2023
Blacksburg, Virginia

Keywords: Transfer Learning, Radio Frequency Machine Learning, Deep Learning
Copyright 2023, Lauren J. Wong
Foundations of Radio Frequency Transfer Learning

Lauren J. Wong

ABSTRACT

The introduction of Machine Learning (ML) and Deep Learning (DL) techniques into modern radio communications system, a field known as Radio Frequency Machine Learning (RFML), has the potential to provide increased performance and flexibility when compared to traditional signal processing techniques and has broad utility in both the commercial and defense sectors. Existing RFML systems predominately utilize supervised learning solutions in which the training process is performed offline, before deployment, and the learned model remains fixed once deployed. The inflexibility of these systems means that, while they are appropriate for the conditions assumed during offline training, they show limited adaptability to changes in the propagation environment and transmitter/receiver hardware, leading to significant performance degradation. Given the fluidity of modern communication environments, this rigidness has limited the widespread adoption of RFML solutions to date.

Transfer Learning (TL) is a means to mitigate such performance degradations by re-using prior knowledge learned from a source domain and task to improve performance on a “similar” target domain and task. However, the benefits of TL have yet to be fully demonstrated and integrated into RFML systems. This dissertation begins by clearly defining the problem space of RF TL through a domain-specific TL taxonomy for RFML that provides common language and terminology with concrete and Radio Frequency (RF)-specific example use-cases. Then, the impacts of the RF domain, characterized by the hardware and channel environment(s), and task, characterized by the application(s) being addressed, on performance are studied, and methods and metrics for predicting and quantifying RF TL performance
are examined. In total, this work provides the foundational knowledge to more reliably use TL approaches in RF contexts and opens directions for future work that will improve the robustness and increase the deployability of RFML.
Foundations of Radio Frequency Transfer Learning

Lauren J. Wong

GENERAL AUDIENCE ABSTRACT

The field of Radio Frequency Machine Learning (RFML) introduces Machine Learning (ML) and Deep Learning (DL) techniques into modern radio communications systems, and is expected to be a core component of 6G technologies and beyond. While RFML provides a myriad of benefits over traditional radio communications systems, existing approaches are generally incapable of adapting to changes that will inevitably occur over time, which causes severe performance degradation. Transfer Learning (TL) offers a solution to the inflexibility of current RFML systems, through techniques for re-using and adapting existing models for new, but similar, problems. TL is an approach often used in image and language-based ML/DL systems, but has yet to be commonly used by RFML researchers. This dissertation aims to provide the foundational knowledge necessary to reliably use TL in RFML systems, from the definition and categorization of RF TL techniques to practical guidelines for when to use RF TL in real-world systems. The unique elements of RF TL not present in other modalities are exhaustively studied, and methods and metrics for measuring and predicting RF TL performance are examined.
For my family – Mom, Dad, Ryan, Grandma, and Paw-Paw – I love you.
Acknowledgments

First, I’d like to thank my advisor, Alan Michaels, for your support throughout both my Masters and PhD. Thank you especially for your flexibility, and for challenging me to make this dissertation the best it could be.

I’d also like to thank my managers at Intel, Sean McPherson and Somdeb Majumdar, for encouraging me to pursue this degree alongside my work, and for acting as a sounding board when I was feeling stuck.

Finally, this work would not have been possible without the help of Braeden Muller and Bill Clark, who both spent endless hours building, debugging, and running the data collection platform in these final months.
Contents

List of Figures xiii

List of Tables xxii

List of Abbreviations xxiii

1 Introduction 1

1.1 Motivation ................................................. 1

1.2 Research Objectives ........................................... 4

1.3 Contributions ................................................ 6

1.3.1 Publications ................................................. 7

1.4 Overview of Chapters ...................................... 10

1.5 Relevance of Research ...................................... 12

2 Radio Frequency Machine Learning 13

2.1 Definition ...................................................... 14

2.2 Applications ................................................... 15

2.2.1 Automatic Modulation Classification (AMC) .......... 15

2.2.2 Specific Emitter Identification (SEI) .................... 17

vii
2.2.3 Signal Detection ........................................... 18
2.2.4 Channel Modeling/Emulation .............................. 18
2.2.5 Positioning/Localization .................................. 19
2.2.6 Spectrum Anomaly Detection .............................. 20
2.3 Data & Dataset Creation ..................................... 20
  2.3.1 Synthetic Data & Datasets ................................. 21
  2.3.2 Captured Data & Datasets ................................. 22
  2.3.3 Augmented Data & Datasets ............................... 23
2.4 Common Approaches & Neural Network Architectures ......... 24
  2.4.1 Multi-layer Perceptrons (MLPs) ........................... 26
  2.4.2 Convolutional Neural Networks (CNNs) .................... 26
  2.4.3 Recurrent Neural Networks (RNNs) ....................... 28
2.5 Challenges & Obstacles to Widespread Deployment ............ 28
  2.5.1 Human-Machine Interaction & End-User Confidence ....... 29
  2.5.2 Real-time Processing Capabilities ....................... 30
  2.5.3 Online & Transfer Learning Techniques .................. 30

3 Transfer Learning for RFML ................................. 32
  3.1 Definitions .................................................. 33
  3.2 Other TL Taxonomies ....................................... 37
3.3 An RFML-Specific Taxonomy & Existing Work ........................................... 38
  3.3.1 Domain Adaptation ................................................................. 41
  3.3.2 Sequential Learning .............................................................. 46
  3.3.3 Multi-task Learning ............................................................... 48
3.4 Discussion .................................................................................. 49

4 An RF TL Experimental Framework .................................................. 50
  4.1 Data .......................................................................................... 51
    4.1.1 Synthetic Data Generation ..................................................... 52
    4.1.2 Captured Data Collection ..................................................... 54
    4.1.3 Dataset Creation ................................................................. 60
  4.2 Model Architectures ..................................................................... 63
    4.2.1 Convolutional Neural Network (CNN) ...................................... 63
    4.2.2 Convolutional Long Short-term Deep Neural Network (CLDNN) . . 65
  4.3 Model Training & Evaluation ....................................................... 66
    4.3.1 Pre-Training ................................................................. 66
    4.3.2 Transfer Learning .............................................................. 67
    4.3.3 Baseline Models ............................................................... 68
    4.3.4 Evaluation ................................................................. 68
  4.4 Real-World Considerations .......................................................... 73
4.4.1 Impacts of Tx/Rx Hardware, CF, and Channel Environment on Base-line Performance ........................................ 74
4.4.2 To Correct or Not to Correct CFO ............................................... 77
4.4.3 Training Time ................................................................. 78
4.4.4 Testing Direct Transfer ......................................................... 78
4.5 Discussion .............................................................................. 80

5 An Analysis of RF Domain Adaptation Behavior 82
5.1 Experiments ........................................................................... 83
  5.1.1 Synthetic Dataset Experiments ........................................... 83
  5.1.2 Captured Dataset Experiments .......................................... 85
5.2 Results .................................................................................. 87
  5.2.1 Head Re-Training vs. Fine-Tuning ..................................... 87
  5.2.2 Synthetic Dataset Experiments .......................................... 89
  5.2.3 Captured Dataset Experiments .......................................... 98
5.3 Discussion .............................................................................. 103

6 An Analysis of RF Sequential Learning Behavior 105
6.1 Experiments ........................................................................... 106
  6.1.1 Synthetic Dataset Experiments .......................................... 106
  6.1.2 Captured Dataset Experiments .......................................... 108
6.2 Results ................................................................. 109

6.2.1 Sequential Learning Across Signal Types ...................... 109
6.2.2 Sequential Learning Across Groups of Txs .................... 111
6.2.3 Sequential Learning For Successive Model Refinement ........ 113
6.2.4 Head Re-training vs. Fine Tuning .............................. 113

6.3 Discussion ............................................................ 120

7 Metrics for Predicting RF TL Performance 122

7.1 Definitions ............................................................ 123

7.1.1 Transferability Metrics ........................................... 123
7.1.2 Dataset Similarity ................................................. 127

7.2 Results ................................................................. 131

7.2.1 Transferability Metrics ........................................... 131
7.2.2 Predicting TL Accuracy Using LEEP & LogME ............... 139
7.2.3 Dataset Similarity ................................................. 141

7.3 Discussion ............................................................ 152

8 Conclusion 154

8.1 Generalized Guidelines for RF TL ................................. 155
8.2 Future Work ......................................................... 158

8.2.1 Additional RF TL Analysis ................................. 158
8.2.2 Algorithmic Development ........................................ 161

Bibliography ................................................................. 163
List of Figures

1.1 The difference between traditional ML and TL. ............... 2

1.2 An outline of the experimental portion of this dissertation. ... 5

2.1 RFML data creation/collection (a) and dataset types (b). ... 21

2.2 An overview of the traditional ML and TL training pipeline. ... 24

2.3 An example MLP with inputs and outputs for AMC and SEI use-cases. ... 26

2.4 An CNN with example input ........................................... 27

2.5 The 1D convolution operation, with a stride of 1 and no padding. ... 27

2.6 A simple RNN with example input. .................................... 28

3.1 The two-dimensional spectrum of “similarity” between source and target domains and tasks. ........................................... 36

3.2 The proposed TL taxonomy for RFML. ................................. 40

4.1 An overview of the synthetic data generation process. .......................... 52

4.2 Front bank of USB hubs on the Transmitter (Tx) host machine with 20 Yard Stick One devices distributed throughout. ................... 55

4.3 Side profile of the Tx host machine with the side panel removed. ........ 55

4.4 The organization structure of data creation and synchronization of transmissions. ........................................... 56
4.5 Overview of process of dataset creation with the BURP platform from start to finish. ................................................................. 57

4.6 An overview of the BURP post-processing stages. ......................... 59

4.7 From each master dataset, subsets containing the desired metadata parameters are selected using configuration files. ......................... 61

4.8 Each burst is ingested and split into segments or examples of desired length within PyTorch. To ensure each example is distinct from one another, a buffer is placed between each example taken from the data file. ...................... 62

4.9 The SNR of the examples in the captured dataset by capture location. ... 63

4.10 The CNN model architecture used for the synthetic AMC experiments, where $n$ is the number of output classes (modulation types) trained. ............ 64

4.11 The CLDNN model architecture used for the captured AMC and SEI experiments, where $n$ is the number of output classes (modulation types or emitter IDs) trained. ..................................................... 65

4.12 A system overview of the model pre-training, TL, and model evaluation processes used in this work. ................................................. 67

4.13 The aggregate classification accuracy of three example NN using the multinomial framework. ......................................................... 71

4.14 The difference between post-transfer top-1 accuracies when trained with and without CFO correction for AMC (a) and SEI (b) models. When the value is positive (to the right of the solid black line), the model trained with CFO correction outperforms the model trained without CFO correction. ............. 76
4.15 The source baseline model accuracy versus the target direct transfer accuracy for AMC and SEI models trained on the captured dataset.  

4.16 The target direct transfer accuracy versus the target post-transfer accuracy, performed using fine-tuning, for AMC and SEI models trained on the captured dataset.  

4.17 The target limited-data (a) and full-data (b) baseline model accuracy versus the target post-transfer accuracy, performed using fine-tuning, for AMC and SEI models trained on the captured dataset.  

5.1 The parameter-of-interest range for each synthetic domain adaptation data subset.  

5.2 The difference between post-transfer top-1 accuracies achieved using head re-training versus fine-tuning for the sweep over (a) Signal-to-Noise Ratio (SNR), (b) Frequency Offset (FO), and (c) SNR + FO.  

5.3 The difference between post-transfer top-1 accuracies achieved using head re-training versus fine-tuning across all domains in the captured dataset for the AMC (a) and SEI use-cases.  

5.4 The post-transfer top-1 accuracy for each source/target dataset pair constructed for the synthetic dataset sweeps over SNR using head re-training (a) and fine-tuning (b).  

5.5 The post-transfer top-1 accuracy for each source/target dataset pair constructed for the synthetic dataset sweeps over FO using head re-training (a) and fine-tuning (b) to perform domain adaptation.
The post-transfer top-1 accuracy for each source/target dataset pair constructed for the synthetic dataset sweeps over SNR + FO using head re-training (a) and fine-tuning (c) to perform domain adaptation.

The difference between post-transfer top-1 accuracy and target limited-data baseline accuracy for the sweep over SNR using head re-training (a) and fine-tuning (b).

The difference between post-transfer top-1 accuracy and target limited-data baseline accuracy for the sweep over FO using head re-training (a) and fine-tuning (b).

The difference between post-transfer top-1 accuracy and target limited-data baseline accuracy for the sweep over SNR + FO using head re-training (a) and fine-tuning (b).

The post-transfer top-1 accuracy for each source/target dataset pair constructed for the captured dataset AMC and SEI experiments using head re-training and fine-tuning.

The post-transfer top-1 accuracy for each source/target dataset pair constructed for the captured dataset AMC and SEI experiments using head re-training and fine-tuning with the rows/columns sorted by limited-data baseline accuracy.

The difference between post-transfer top-1 accuracy and target limited-data baseline accuracy for each source/target dataset pair constructed for the captured dataset AMC and SEI experiments using head re-training and fine-tuning.
5.13 The post-transfer top-1 accuracy across changing groups of Txs for the captured dataset AMC use-case using head re-training and fine-tuning. .......................... 101

5.14 The difference between post-transfer top-1 accuracy and target limited-data baseline accuracy across changing groups of Txs for the captured dataset AMC use-cases using head re-training and fine-tuning. .......................... 101

5.15 The SNR for the top and bottom three performing domains for the (a) AMC use-case and (b) SEI use-case. .......................... 102

6.1 The modulation schemes in each data-subset in the Synthetic Sequential AMC experiment. .......................... 106

6.2 The modulation schemes in each data-subset in the Synthetic Model Refinement AMC experiment. .......................... 107

6.3 The modulation schemes in each data-subset in the Captured Model Refinement AMC experiment. .......................... 108

6.4 The post-transfer top-1 accuracy for each source/target dataset pair constructed for the Synthetic Sequential AMC experiment using head re-training (a) and fine-tuning (b). .......................... 110

6.5 The difference between post-transfer top-1 accuracy and target limited-data baseline accuracy for the Synthetic Sequential AMC experiment using head re-training (a) and fine-tuning (b). .......................... 110

6.6 The post-transfer top-1 accuracy for each source/target dataset pair constructed for the Captured Sequential SEI experiment using head re-training (a) and fine tuning (b). .......................... 112
6.7  The difference between post-transfer top-1 accuracy and target limited-data baseline accuracy for the Captured Sequential SEI experiment using head re-training (a) and model fine-tuning (b). ........................................... 112

6.8  The post-transfer top-1 accuracy for each source/target dataset pair constructed for the Synthetic Model Refinement AMC experiment using head re-training (a) and fine-tuning (b). .................................................. 114

6.9  The difference between post-transfer top-1 accuracy and target baseline accuracy for Synthetic Model Refinement AMC using head re-training (a) and fine-tuning (b). .................................................. 114

6.10 The post-transfer top-1 accuracy for each source/target dataset pair constructed for the Captured Model Refinement AMC experiment using head re-training (a) and fine tuning (b). ............................... 115

6.11 The difference between post-transfer top-1 accuracy and target limited-data baseline accuracy for the Captured Model Refinement AMC experiment using head re-training (a) and model fine-tuning (b). ................................. 115

6.12 The post-transfer top-1 accuracy for each source/target dataset pair constructed for the Captured Model Refinement SEI experiment using head re-training (a) and fine tuning (b). ............................... 116

6.13 The difference between post-transfer top-1 accuracy and target limited-data baseline accuracy for the Captured Model Refinement SEI experiment using head re-training (a) and model fine-tuning (b). ................................. 116
6.14 The difference between post-transfer top-1 accuracies achieved using head re-training versus fine-tuning for (a) the Synthetic Sequential AMC and (b) Synthetic Model Refinement AMC experiments. 117

6.15 The difference between post-transfer top-1 accuracies achieved using head re-training versus fine-tuning for (a) the Captured Model Refinement AMC, (b) Captured Sequential SEI, and (c) Captured Model Refinement SEI experiments. 118

7.1 The LEEP and LogME scores versus post-transfer top-1 accuracy for the synthetic sweep over SNR, FO, and SNR+FO. 132

7.2 The LEEP and LogME scores versus post-transfer top-1 accuracy for the captured domain adaptation experiments with tanh function fits. 133

7.3 The LEEP and LogME scores versus post-transfer top-1 accuracy for the synthetic sequential learning experiments. 134

7.4 The LEEP and LogME scores versus post-transfer top-1 accuracy for the captured sequential learning experiments. 135

7.5 The LEEP versus LogME scores for the synthetic domain adaptation (a) and sequential learning (b) experiments. 137

7.6 The LEEP versus LogME scores for the captured domain adaptation AMC (a) and SEI (b) experiments. 137

7.7 The LEEP versus LogME scores for the captured sequential learning experiments. 138
7.8 The error in the predicted post-transfer accuracy using a linear fit to the Log Expected Empirical Prediction (LEEP) scores (x-axis) and Logarithm of Maximum Evidence (LogME) scores (y-axis) for the synthetic domain adaptation experiments. .................................................. 142

7.9 The error in the predicted post-transfer accuracy using a linear fit to the LEEP scores (x-axis) and LogME scores (y-axis) for the synthetic sequential learning experiments. .................................................. 143

7.10 The error in the predicted post-transfer accuracy using a Hyperbolic Tangent (tanh) fit to the LEEP scores (x-axis) and LogME scores (y-axis) for the captured domain adaptation experiments. .................................................. 143

7.11 The error in the predicted post-transfer accuracy using a tanh fit to the LEEP scores (x-axis) and LogME scores (y-axis) for the captured sequential learning experiments. .................................................. 144

7.12 Similarity across the synthetic datasets with varying (a) SNR, (b) FO, and (c) SNR and FO. .................................................. 145

7.13 Similarity across the synthetic datasets containing only linear, only frequency-shifted, and only analog modulation schemes, and a mixture of modulation types (a) and the synthetic datasets with a single modulation scheme added/removed (b). .................................................. 146

7.14 Similarity across the captured datasets with varying domains (a) and tasks (b) for the AMC use-case. .................................................. 146

7.15 The difference between the accuracy of TL models versus baseline models as a function of the proposed metric for the synthetic dataset. .................................................. 148
7.16 The difference between the accuracy of TL models versus baseline models as a function of the proposed metric for the captured data AMC use-case.

7.17 The difference between the accuracy of TL models versus baseline models as a function of the proposed metric for the Synthetic Model Refinement AMC experiment.

7.18 Dataset similarity as a function of dataset size and the resultant number of bins per histogram used in each $\chi^2$ test for the first column of source/target dataset pairs shown in Figure 7.13b, averaged over 10 iterations.
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>An overview of the dataset and model types used in popular RFML works.</td>
<td>16</td>
</tr>
<tr>
<td>3.1</td>
<td>Example RFML Domain Elements and Tasks.</td>
<td>34</td>
</tr>
<tr>
<td>3.2</td>
<td>The settings which describe points (a)-(i) on the two-dimensional spectrum of “similarity” between source and target domains and tasks shown in Fig. 3.1.</td>
<td>35</td>
</tr>
<tr>
<td>3.3</td>
<td>Representative examples for TL settings in RFML.</td>
<td>39</td>
</tr>
<tr>
<td>4.1</td>
<td>The generation parameters and signal types included in the synthetic datasets.</td>
<td>53</td>
</tr>
<tr>
<td>4.2</td>
<td>The full-data and limited-data baseline top-1 accuracies for the AMC models trained on data transmitted by different non-overlapping groups of Tx.</td>
<td>74</td>
</tr>
<tr>
<td>4.3</td>
<td>The full-data and limited-data baseline top-1 accuracies for the AMC and SEI models trained on data captured by different collection nodes.</td>
<td>75</td>
</tr>
<tr>
<td>4.4</td>
<td>The full-data and limited-data baseline top-1 accuracies for the AMC and SEI models trained on data captured at different Center Frequencies (CFs).</td>
<td>75</td>
</tr>
<tr>
<td>4.5</td>
<td>The full-data and limited-data baseline top-1 accuracies for the AMC and SEI models trained on data collected at different locations.</td>
<td>76</td>
</tr>
<tr>
<td>4.6</td>
<td>Approximate training time for each model trained in this dissertation.</td>
<td>77</td>
</tr>
<tr>
<td>7.1</td>
<td>The instantaneous time-domain features used in this work, where $a(t)$, $\varphi(t)$, and $f_N(t)$ are the instantaneous amplitude, phase, and frequency of the example $s(t)$, respectively.</td>
<td>130</td>
</tr>
</tbody>
</table>
7.2  The frequency with which the proposed method constructed using LEEP and LogME agree in over/under predicting post-transfer accuracy.  

7.3  The Pearson’s $r$ correlation coefficient between difference between the accuracy of TL models versus baseline models and the proposed dataset similarity metric for all AMC experiments.
List of Abbreviations

AE  Autoencoder

AM  Amplitude Modulation

AMC  Automatic Modulation Classification

APSK  Amplitude and Phase-Shift Keying

ASK  Amplitude-Shift Keying

AWGN  Additive White Gaussian Noise

BPSK  Binary Phase-Shift Keying

CLDNN  Convolutional Long Short-term Deep Neural Network

CNN  Convolutional Neural Network

CF  Center Frequency

CFO  Carrier Frequency Offset

CR  Cognitive Radio

CV  Computer Vision

DL  Deep Learning

DSA  Dynamic Spectrum Access

DSB  double-sideband
DSBSC  double-sideband suppressed-carrier
DSP  Digital Signal Processing
DTW  Dynamic Time Warping
DQN  Deep Q-Networks
FFT  Fast Fourier Transform
FID  Frechet Inception Distance
FIR  Finite Impulse Response
FM  Frequency Modulation
FO  Frequency Offset
FPGA  Field-Programmable Gate Array
FSK  Frequency-Shift Keying
GAN  Generative Adversarial Network
GBC  Gaussian Bhattacharyya Coefficient
GFSK  Gaussian Frequency-Shift Keying
GMSK  Gaussian Minimum-Shift Keying
GPU  Graphical Processing Unit
GRU  Gated Recurrent Unit
IEEE  Institute of Electrical and Electronics Engineers
IIR  Infinite Impulse Response
IoT  Internet-of-Things

IQ  In-phase and Quadrature

JC-NCE  Joint Correspondences Negative Conditional Entropy

JS divergence  Jenson-Shannon Divergence

KL divergence  Kullback-Leibler Divergence

LSB  lower-sideband

LSTM  Long Short-Term Memory

LEEP  Log Expected Empirical Prediction

LogME  Logarithm of Maximum Evidence

ML  Machine Learning

MLP  Multi-layer Perceptron

MMD  Maximum Mean Discrepancy

MSE  Mean Squared Error

MSK  Minimum-Shift Keying

NB  narrowband

NCE  Negative Conditional Entropy

NLL  Negative Log-Likelihood

NLP  Natural Language Processing

NN  Neural Network
OOK  On-Off Keying

OQPSK  Offset Quadrature Phase-Shift Keying

OTCE  Optimal Transport-based Conditional Entropy

PCA  Principal Component Analysis

PFA  Principal Feature Analysis

PSK  Phase-Shift Keying

QAM  Quadrature Amplitude Modulation

QPSK  Quadrature Phase-Shift Keying

RF  Radio Frequency

RFML  Radio Frequency Machine Learning

RL  Reinforcement Learning

RNN  Recurrent Neural Network

RSS  Received Signal Strength

Rx  Receiver

SEI  Specific Emitter Identification

SGD  Stochastic Gradient Descent

SINR  Signal-to-Interference-Plus-Noise Ratio

SNR  Signal-to-Noise Ratio

STFT  Short-Time Fourier Transform
SVM  Support Vector Machine

tanh  Hyperbolic Tangent

TL  Transfer Learning

Tx  Transmitter

UAV  Unmanned Aerial Vehicle

USB  upper-sideband

WB  wideband
Chapter 1

Introduction

1.1 Motivation

The past decade has seen Machine Learning (ML) and Deep Learning (DL) techniques successfully applied to a wide variety of modalities including imagery, audio, and text, yielding commercialized Computer Vision (CV), speech recognition, and Natural Language Processing (NLP) algorithms. More recently, the application of ML and DL techniques to wireless communications tasks has yielded state-of-the-art spectrum awareness [1], cognitive radio [2], and networking algorithms [3]. Such Radio Frequency Machine Learning (RFML) techniques boast increased performance and flexibility with less pre-processing and reliance on pre-defined expert features compared to traditional signal processing techniques [4, 5, 6]. However, RFML techniques are only beginning to see widespread adoption in commercial and military systems, largely due to a lack of research that considers real-world challenges such as human-machine interaction and end-user confidence, achieving real-time processing capabilities, and online and TL techniques [7].

The vast majority of RFML works train ML/DL models from random initialization using supervised learning techniques, thereby assuming the availability of a large corpus of labeled training data representative of the expected deployment environment [8]. In practice, this assumption inevitably breaks down due to hardware variations, as well as non-linear and non-repeatable channel effects, and as a result, performance degrades severely. For example,
Figure 1.1: The difference between traditional ML (a), in which a new model is trained on a each domain/task pairing from random initialization, and TL (b), in which prior knowledge learned on one domain/task is used to support performance on a second domain and/or task where less (or no) labeled data is available. Traditional ML is ideal when large amounts of labeled training data is available for the target domain/task. However, models are likely to degrade over time, as the domain/task changes. TL is beneficial when little-to-no labeled training data is available for the target domain/task, but large amounts of labeled training data for a similar source domain/task is available. TL can also be used to make small updates to models as the domain/task changes over time.

Preliminary results given in [9] showed that the performance of CNN and Long Short-Term Memory (LSTM)-based AMC algorithms trained on data from a single Tx/Receiver (Rx) pair not only varied significantly across different Tx/Rx pairs, but also dropped by as much as 8% when tested on data captured from other Tx/Rx pairs even when augmentations were applied to improve performance. Work in [10] showed that Convolutional Neural Network (CNN) and LSTM-based Automatic Modulation Classification (AMC) models trained using synthetic data performed poorly on captured test data, achieving only 50% of the accuracy of models trained using captured or augmented data. Furthermore, results in [11] showed that a CNN-based Specific Emitter Identification (SEI) model learned to correlate channel distortions with Txs, rather than learning the characteristics of the Txs themselves, and consequently performed poorly under different channel conditions.

Transfer Learning (TL) aims to overcome these obstacles by utilizing prior knowledge gained
from a *source* domain and task to improve performance of a model for a “similar” *target* domain and task, when compared to training only on the target domain and task from random initialization (Figure 1.1). In the context of RFML, the domain consists of the RF hardware and channel environment(s) and the task comprises the application(s) being addressed, a topic discussed further in Chapter 3. Therefore, TL provides an avenue for

- Increasing performance with reduced captured training data – While TL methods are beneficial in a wide variety of learning scenarios, TL shines when sufficient training data is not available in the target domain, yet similar source data is available from which knowledge of the target domain/task can be gleaned.

- Yielding high performing models across a wide variety of hardware platforms and channel conditions while minimizing (re)training time.

For example, large captured training datasets have been shown to yield the greatest performance, but require several orders of magnitude more time to create when compared to synthetic and augmented datasets [12]. TL enables comparable performance with less captured training data by transferring models trained on large amounts of near-distribution synthetic, augmented, and/or captured data to the smaller in-distribution captured dataset. Additionally, unlike in CV or NLP, behavior learned on one RF platform will be distinctly impacted by the Tx and Rx hardware, and will therefore vary from platform to platform. Channel effects including time of day, multi-path profile, and noise variations, also impact learned behavior and can be difficult to model. TL provides the capability to tailor a model to specific platforms and channel environments, facilitating reliable long-term performance.

Existing works have applied TL techniques developed for CV and NLP applications to the Radio Frequency (RF) space, but provide very little understanding of RF TL behavior and performance. While parallels can be drawn between RFML and other modalities in which TL
has been employed successfully, borrowing such approaches yields no guarantee of success. For example, while it would seem that changes in Tx/Rx hardware (the RF platform) would mirror changes in camera for CV algorithms, changing between cameras that all capture at the same resolution does not significantly affect performance of a CV model [13]. Meanwhile, changes in the Tx/Rx hardware has been shown to materially impact performance, despite using the same settings [9]. Without understanding how TL techniques perform under various RF conditions, we are unable to effectively utilize the technology which will aid in seeing the widespread adoption of RFML systems. More specifically, designing TL algorithms for the RFML space will first require a fundamental understanding of how both the channel environment and platform variations impact learned behavior and inhibit or facilitate transfer. This dissertation aims to bridge this gap through exhaustive experimentation and performance evaluation using synthetic and captured data, across two different use-cases, and with varying hardware and channel conditions, yielding generalized guidelines and best practices for performing RF TL.

1.2 Research Objectives

In order to fully reap the benefits of TL in the context of RFML, we must thoroughly understand how TL techniques perform under various RF-specific conditions. As such, this dissertation seeks to begin addressing the following questions:

1. What are the underlying system parameters affecting the effectiveness of transferring learned behaviors?

2. What are general guidelines/best practices for RF TL?

3. What are some of the unique challenges and limitations of RF TL?
Figure 1.2: An outline of the experimental portion of this dissertation, organized according to the RF-specific TL taxonomy presented in Chapter 3.

4. Can TL performance reasonably be predicted?

To begin, the RF specific TL taxonomy shown in Figure 1.2 is presented and discussed extensively in Chapter 3, providing a framework for comparing and contrasting existing works and underscoring the unique facets of RFML that may warrant deliberate TL algorithm design and yield significant performance gains. Then, the results of an exhaustive experimental examination of TL performance is presented, with particular emphasis on domain adaptation and sequential learning behavior, in which the TL methods (i.e. training hyperparameters) are fixed and the RF domain or task is varied. Specifically, performance trends are examined across synthetic and captured datasets, two different use-cases, and two different Neural Network (NN) architectures to identify how changes in propagation environment, Tx/Rx hardware, and Center Frequency (CF) independently and jointly impact RF TL be-
behavior, generalizing over use-case and Neural Network (NN) architecture. Then, metrics and methods for quantifying dataset similarity and transferability, as well as source model selection and predicting TL performance are considered. The results of these experiments together yield a summary of generalized guidelines for successful RF TL.

1.3 Contributions

In total, the contributions of this dissertation are as follows:

- A domain-specific TL taxonomy for RFML (Chapter 3),
- A survey of existing RF TL research (Chapter 3),
- An open-source synthetic RF dataset, open-source synthetic RF dataset generation tool, and soon to be open-source captured RF dataset. The public release of these datasets and codebase increases the transparency and repeatability of the results herein (Chapter 4),
- A systematic evaluation of RF domain adaptation performance (Chapter 5),
- A systematic evaluation of RF sequential learning performance (Chapter 6),
- An examination of how well existing modality agnostic transferability metrics, LEEP and LogME, perform on RFML models and datasets (Chapter 7),
- A method for using any transferability metric to predict post-transfer accuracy, within a confidence interval, and without further training (Chapter 7),
- A novel dataset similarity metric, based on expert-defined features and $\chi^2$ tests, which intuitively quantifies the notion of similarity between RF datasets (Chapter 7),
• A set of generalized guidelines and best practices for successful RF TL distilled from the results of the experiments performed herein (Chapter 8),

• An outline of suggested experiments for addressing several remaining open questions of RF TL behavior (Chapter 8).

1.3.1 Publications

The research presented in this dissertation has resulted in the publication of 8 journal papers, 1 book chapter, 1 conference paper, and 2 datasets spanning a range of topics including a survey of RFML, novel SEI approaches, and surveys of RF TL and behavior analysis. These publications, outlined below, have accumulated over 350 citations, with an h-index of 8 and an i10-index of 7, emphasizing the early impact these publications have had on the research community.

The following journal and conference publications present novel and practical CNN-based SEI approaches that utilize only raw In-phase and Quadrature (IQ) data as input, and show the utility of such approaches in an Internet-of-Things (IoT) setting:


The following book chapter describes a multinomial distribution-based predictive approach to aggregating the results of discrete NN outputs that is used in the captured data experiments throughout this dissertation:


This journal publication provides a holistic overview and survey of prior works related RFML applications, dataset creation, security, trust and assurance, and operational considerations with particular attention paid to RF modality specific considerations which not present in fields such as CV or NLP:


The following journal paper details the construction and configuration of the “BURP Machine” which was used to create the captured RF dataset used in this work:


The synthetic and captured RF datasets created for and used in this work are publicly
available on Institute of Electrical and Electronics Engineers (IEEE) DataPort, and the synthetic RF dataset generation tool developed for this work is open-source on GitHub, allowing other researchers to reproduce and extend the work performed herein:


The following journal publication presents the RF-specific TL taxonomy providing the framework for comparing and contrasting existing works. The framework is then put into practice when surveying existing works:


The following conference and journal publications present the experimental results of Chapters 5 - 6:


1.4 Overview of Chapters

This dissertation is organized as follows:

Chapter 1 has provided an introduction to this work, motivating the use of TL in RFML, highlighting the need for increased understanding of RF TL performance across changes in environmental conditions and hardware imperfections. This chapter has also introduced four key research questions that are central to the work presented herein, and has listed the contributions of this dissertation.

As this dissertation focuses on the application of TL techniques to RFML, Chapter 2 provides key background knowledge on RFML, including the definition of RFML, as well as the use-cases or applications, types of data, and DL architectures found in the literature. This chapter provides a review of the literature on state-of-the-art RFML, highlights the challenges unique to the RF modality, and builds the foundation necessary to understand the decisions made in the design of the experimental framework presented in Chapter 4.

Chapter 3 focuses on TL, clearly defining the problem space as it relates to the RF modality. Existing TL taxonomies are described, and a domain-specific TL taxonomy for RFML is presented, providing common language and terminology for future RF TL works. Each component of the RF TL taxonomy is described in detail, with concrete examples, and the limited body of existing work is surveyed. This chapter concludes with discussion of the most apparent directions for future work in RF TL given the literature surveyed within,
1.4. Overview of Chapters

leading to the experimental study conducted for this dissertation.

Chapter 4 presents the experimental framework used throughout this dissertation to analyze TL performance, generalizing over use-case and NN architecture. The two datasets created/collected for this work are described in detail, as well as the two NN architectures used and the model pre-training and TL pipelines. Further motivating this work and providing context for the design decisions made, Chapter 4 concludes with initial results on the impacts of Carrier Frequency Offset (CFO) correction on RFML and RF TL performance, the baseline performance of the AMC and SEI models trained across varying CFs, channels, and Tx/Rx variations, and the expected performance of these baseline models in changing RF domains both with and without TL.

Chapters 5 and 6 systematically evaluate the impact of changes in the propagation environment and Tx/Rx hardware on domain adaptation and sequential learning performance. Performance is quantified using post-transfer top-1 accuracy across changes in Signal-to-Noise Ratio (SNR), Frequency Offset (FO), Center Frequency (CF), real-world channels, modulation types, Tx ID, and Rx ID for both AMC and SEI use-cases and using both synthetic and captured data. The results presented in these chapters discuss real-world considerations such as the impacts of domain similarity and “difficulty” on performance, the relative challenge of overcoming changes in environment versus changes in Tx/Rx hardware, as well as the pros and cons of using different TL techniques.

Chapter 7 focuses on metrics for predicting RF TL behavior without further training. Using post-transfer top-1 accuracy as the ground truth measure of “transferability,” this chapter examines how existing transferability metrics (LEEP and LogME) and a novel RF dataset similarity metric correlate with post-transfer top-1 accuracy. Additionally, a method is proposed using LEEP and LogME to predict post-transfer accuracy, within a confidence interval.
Finally, Chapter 8 consolidates the results of the previous three chapters into generalized and actionable guidelines for RF TL. Where additional research is needed, this chapter suggests and outlines specific experiments for future work.

1.5 Relevance of Research

To date, the primary area of research in RFML has focused on providing novel solutions to spectrum awareness and cognitive radio tasks. Meanwhile, only limited attention has been paid to the impacts of the data on learned behavior, vulnerabilities of RFML in adversarial contexts, and the requirements for deploying these algorithms in real-world applications. Without building greater flexibility into RFML-based systems to adjust their responses to observed stimuli, such systems will only be able to support signal and environments encountered previously. Ultimately, these limitations have hampered the widespread adoption of RFML algorithms thus far.

One such requirement for real-world deployment is for trained models to maintain performance over time, despite changing hardware and channel conditions. TL provides a method for adapting pre-trained models to new hardware and channel conditions, as well as to new use-cases, but is not sufficiently understood to be utilized routinely and reliably in RF contexts. Towards this goal, this dissertation clearly defines the problem space in the context of RFML through the taxonomy presented in Chapter 3, evaluates TL performance as a function of RF-specific parameters and characteristics, and presents metrics and methods for predicting and quantifying RF TL performance. The results presented herein increase our understanding of RF TL performance, as well as our understanding of RFML more broadly, with regards to how environment, hardware, and use-case impact learned behavior.
Chapter 2

Radio Frequency Machine Learning

Throughout this dissertation, some level of familiarity with RFML is assumed, including the motivations for RFML use, the ML/DL techniques typically employed, and some of the unique challenges of RFML not present in fields such as CV and NLP. This chapter provides this foundation, and sets the scene for the decisions made in the design of the experimental framework discussed in Chapter 4.

To start, the term RFML is more rigorously defined in Section 2.1, narrowing the scope of works discussed herein. Then, Section 2.2 highlights the relevant RFML applications found in the literature, which include AMC and SEI; the experimental use-cases examined in Chapters 5 - 6. In Section 2.3, the types of RFML datasets used in existing work are described, and the impact of real-world effects on RF data are discussed, motivating further research in RF TL. Section 2.4 provides a brief overview of the DL architectures and approaches typically used in existing RFML works, touching on how architectures and approaches were chosen for this work. Finally, this chapter concludes with a discussion of a few of the key challenges and areas of future research needed to mature RFML for deployment in Section 2.5, including TL techniques.

This chapter is an adaptation of the following publication:

2.1 Definition

In recent years, Deep Learning (DL) algorithms have been utilized in the wireless communications domain for facilitating spectrum situational awareness applications such as signal detection, signal parameter estimation, AMC, and SEI. Given the initial successes in these areas, among others in the wireless communications domain, DL is considered a transformative technology in the upcoming 5G standard and is expected to be a core component of 6G technologies and beyond [14].

While the term RFML has been used in the literature to loosely describe any application of ML to the RF domain, RFML systems were first defined as systems [15]:

- That utilize autonomous feature learning from raw data that can “learn the characteristics used to identify and characterize signals,"
- That are used to detect, identify, and recognize signals-of-interest,
- That are able to autonomously configure the RF sensor or communications platform to be most effective in changing communications environments, and
- That are “Able to digitally synthesize virtually any possible waveform.”

These guiding principles are used to narrow the scope of the subject matter examined herein, and focus discussions and the literature review undertaken on techniques aiming to reduce the amount of expert-defined features and prior knowledge needed for the intended application. More specifically, this discussion focuses on works that utilize time-domain data at baseband,
or transforms thereof (i.e. Fast Fourier Transforms (FFTs), Short-Time Fourier Transforms (STFTs)), as input to ML techniques, while works utilizing or deriving pre-defined expert features as input to classical ML methods are drawn upon only for context.

2.2 Applications

The following subsections overview and survey common RFML applications found in the literature. An overview of the works discussed herein, including training data types and model types, is given in Table 2.1. It should be noted that the works cited are not exhaustive, and rather serve as quality examples of work in the area.

2.2.1 Automatic Modulation Classification (AMC)

One of the earliest, and perhaps the most researched, applications of RFML for spectrum situational awareness is that of AMC, likely due to the historical success of ML techniques on classification tasks across modalities. Traditional AMC techniques typically consist of two signal processing stages: feature extraction and pattern recognition [41]. The feature extraction stage has typically relied on the use of so-called “expert features” in which a human domain-expert pre-defines a set of signal features that allow for statistical separation of the modulation classes of interest, examples of which can be found in [41]. These expert-defined signal features are extracted from the raw received signal during a potentially time intensive and computationally expensive pre-processing stage, then used as input to a pattern recognition algorithm, which may consist of decision trees, Support Vector Machines (SVMs), NNs, among many others.

RFML-based approaches aim to replace the human intelligence and domain expertise re-
<table>
<thead>
<tr>
<th>Model Type</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>[30]</td>
</tr>
<tr>
<td>CNN</td>
<td>[31]</td>
</tr>
<tr>
<td>RNN</td>
<td>[32]</td>
</tr>
<tr>
<td>GAN</td>
<td>[33]</td>
</tr>
<tr>
<td>Other</td>
<td>[34]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application</th>
<th>Dataset Type</th>
<th>Model Type</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomaly Detection</td>
<td>X</td>
<td>X</td>
<td>[16]</td>
</tr>
<tr>
<td>Positioning/Localization</td>
<td>X</td>
<td>X</td>
<td>[35]</td>
</tr>
<tr>
<td>Channel Modeling/Emulation</td>
<td>X</td>
<td>X</td>
<td>[36]</td>
</tr>
<tr>
<td>Positioning/Localization</td>
<td>X</td>
<td>X</td>
<td>[37]</td>
</tr>
<tr>
<td>SEI</td>
<td>X</td>
<td>X</td>
<td>[38]</td>
</tr>
<tr>
<td>Signal Detection</td>
<td>X</td>
<td>X</td>
<td>[39]</td>
</tr>
<tr>
<td>Signal Detection</td>
<td>X</td>
<td>X</td>
<td>[40]</td>
</tr>
<tr>
<td>Real Synthetic Augmented</td>
<td>X</td>
<td>X</td>
<td>[41]</td>
</tr>
</tbody>
</table>

Table 2.1: An overview of the dataset and model types used in popular RFML works.
required to identify and characterize these features using deep NNs, such as CNNs and RNNs, to both blindly and automatically identify separating features and classify signals of interest, with minimal pre-processing and less a priori knowledge [4, 16, 17, 18, 19, 20, 21, 22, 42].

Given the significant research in RFML-based modulation classification, it can be argued that AMC is one of the most mature fields in RFML, and has been deployed in real-world products [43]. Given the maturity and popularity of the AMC and given the ability to perform AMC using both synthetic and captured datasets, AMC is used as one of the primary experimental use-cases in this dissertation.

### 2.2.2 Specific Emitter Identification (SEI)

The goal of Specific Emitter Identification (SEI) or RF Fingerprinting is to identify the unique Tx responsible for sending a signal-of-interest [44]. Slight but consistent differences between Txs, such as IQ imbalances, amplifier non-idealities, and other imperfections caused during the manufacturing process [30] make SEI possible. These differences not only exist between Tx brands and models, but amongst Txs of the same brand and model, which may even have been manufactured side-by-side. Additionally, work presented in [45] showed geographical differences including propagation channels and angle of arrival to have a dramatic effect on SEI performance as well.

Given the vast number of existing devices, each exhibiting nearly imperceptible differences from another, SEI in particular has benefited greatly from the advent of RFML [6, 30, 31, 46]. While traditional SEI techniques have focused on the difficult and laborious task of defining expert features to distinguish between Txs [47], recent RFML-based solutions have used CNNs to learn the discriminating features for identifying transmitters more reliably than the hand-crafted features, and have shown the ability to identify unknown Txs [6, 30, 46].
SEI is used as the secondary experimental use-case in this dissertation, and is only performed using captured datasets, as the differences between Txs cannot be sufficiently modelled in a synthetic dataset.

2.2.3 Signal Detection

Another area of spectrum situational awareness seeing a particular increase in the RFML literature is signal detection [1, 23, 25]. Most often, signal detection is discussed in the context of spectrum sensing as a step in identifying a specific or primary user of the spectrum [24, 26, 27, 28, 29], and is traditionally performed using various energy detection methods and/or matched filtering.

Spectrogram-based signal detection is prime example of a setting in which an image processing techniques have directly been applied to solve an RFML problem. More specifically, in [1, 23], the raw IQ samples were converted into spectrum waterfall plots to allow the spectrum information to be viewed as an image on a time-frequency plane. This has allowed a rich class of existing image processing techniques to be applied directly to perform near real-time signal detection in positive SNR environments. Additional work in [29] explores the use of Generative Adversarial Network (GAN) networks to improve the signal detection performance of systems using compressive sensing.

2.2.4 Channel Modeling/Emulation

The channel plays a defining role in the performance of RFML systems, a topic further discussed in Section 2.3. As a consequence, including realistic channel effects, captured or simulated, into the training of RFML systems is critical to achieving top performance. In the case that sufficient data can not be captured, channel modeling is a critical component
of creating realistic simulations of RF systems.

Traditionally, channel modeling requires understanding the multi-path propagation effects of a wireless channel and stochastically recreating those characteristics using mathematical approximations during simulation. However, such approaches are often computationally expensive [48]. The area of RFML-based channel modeling and/or emulation is currently limited, but continues to grow as the need for data grows. For example, in [32, 33], a ML-based channel “stand-in” is used, which allows for channel emulation within an end-to-end RFML training routine. Alternatively, in [34], the goal is channel translation, where signal captures collected in one channel environment are augmented to resemble a different channel environment.

### 2.2.5 Positioning/Localization

Positioning and localization play a crucial role in both military and commercial communications. For example, as the quantity of consumer-focused wireless devices continue to grow, positioning and localization become increasingly useful in emergency and safety applications, such as search and rescue operations [35, 36].

Traditionally, localization techniques have relied on expert-defined features such as received signal strength [37, 49]. However, in recent years a more rich set of RF measurements including channel transfer functions, frequency coherence functions, and channel state information have been used [36, 38]. While channel state information has been used to reach state-of-the-art and cm-level accuracy on indoor positioning tasks [36], little-to-no work has made progress towards performing localization using raw RF data.
2.2.6 Spectrum Anomaly Detection

An emerging RFML application area is that of anomalous event detection where DL models are used to learn a baseline environment and subsequently detect/classify deviations from this baseline (so-called anomalies). An example of this budding area of research can be found in [39], where RF spectrum activities are monitored and analyzed using deep predictive coding NNs to identify anomalous wireless emissions within spectrograms. Similarly, in [40], the authors utilized recurrent neural predictive models to identify anomalies in raw IQ data. Such approaches also show promise as methods for detecting adversarial attacks or identifying out-of-distribution examples.

2.3 Data & Dataset Creation

In any application of ML, representative and well-labeled datasets are of critical importance for training and/or evaluation. RFML is no different. For RFML, observations in the dataset take the form of time-domain samples of an RF signal, most commonly in complex baseband format with IQ notation, referred to as raw IQ data. The data used in existing RFML works can be categorized into one of three types, shown in Figure 2.1: simulated, captured, and augmented [50]. This following subsections discuss these types of data used in this work and other RFML works, how these types of datasets affect the resulting RFML model, and real-world effects that must be considered when developing RFML datasets. A more descriptive comparison of quality and quantity for these three dataset types to achieve varying levels of AMC performance can be found in [12].
2.3. DATA & DATASET CREATION

(a) A radio communications system indicating where simulated/synthetic, real-world/captured, and augmented data is created/collected.

(b) The types of data used in RFML.

Figure 2.1: RFML data creation/collection (a) and dataset types (b).

2.3.1 Synthetic Data & Datasets

Synthetic datasets, also known as simulated datasets, refer to datasets composed of synthetically generated data, in which the Tx, channel, and Rx are all modeled in interconnected software and/or hardware systems. Synthetic datasets are the most straightforward to compile and label, and can be good analogs for captured RFML datasets, if carefully crafted and known models exist for the intended deployment environment. More specifically, the same equations and processes used to transmit waveforms in real RF systems can be used directly
in simulation [42], unlike in image processing [51], and for simplistic environments, mathematical models can be used to reasonably describe common degradations such as additive interference, channel effects, and Tx/Rx imperfections. Publicly available toolsets such as GNU Radio [52], liquid-dsp [53], and MATLAB [54] provide a low barrier to entry. As a result, synthetic datasets are particularly well-suited to initial development, are the most commonly used type of data in current RFML literature, and are used in the first set of experiments herein. However, models trained on simulated datasets are typically insufficient when applied to captured data, as is seen during real-world deployment [12].

2.3.2 Captured Data & Datasets

In contrast, captured datasets, also known as collected or real-world datasets, contain signals that have been transmitted over a wireless channel. Therefore, captured data includes all of the degradations that are of concern in practical RF situations [11]. These real-world effects can generally be categorized as either hardware variations or channel effects, both of which can significantly impact RFML performance if not considered when developing the training, validation, and test datasets.

Hardware variations refer to the variances between Tx and Rx hardware platforms and the resulting impact on the received waveform. More specifically, different Tx and Rx pairs distort waveforms from the ideal to varying degrees as a result of manufacturing variations, environmental operating conditions (i.e. temperature), and access to supporting devices like reference oscillators. These distortions take the form of non-linearities, additive noise, timing offsets, FOs, phase offsets, sample rate mismatches, and/or amplitude offsets, all of which may be time varying. Distortions to the waveform caused by the Tx may be a parameter-of-interest, as in the case of SEI, or can be a confounding variable, as is generally the case
2.3. DATA & DATASET CREATION

in AMC settings. In the latter case, it is important to train over many Tx/Rx pairs, to generalize over any Tx/Rx variations such as IQ imbalance, SNR, and/or synchronization errors (i.e., time, frequency, and phase offsets) [17, 49, 55].

Signal propagation and/or channel effects add noise and further degrade the signal-of-interest. While most RFML works using synthetic data assume an Additive White Gaussian Noise (AWGN) channel, real-world channels have time-varying, often colored spectra impacted by uncontrolled RF interference sources such as other signals, relative motion between platforms, physical objects, impulsive noise (i.e., lightning), and non-linear effects associated with bursty packet transmissions.

Captured data is critical for test and evaluation prior to real-world deployment, and is therefore used in the second set of experiments herein. The improved realism of captured data also reduces end-user resistance and doubt surrounding the system. However, captured data requires significant labor and resources to both gather sufficiently diverse captures for producing a training and/or evaluation datasets and to label it correctly [17]. Further discussion on the challenges of collecting real-world RF dataset for use in RFML algorithms can be found in [50].

2.3.3 Augmented Data & Datasets

Augmented datasets combine simulated and captured data to increase the quantity of data available for training or to incorporate more realism into a dataset, and aim to minimize the limitations of synthetic datasets (i.e., real-world model accuracy) while using small amounts of captured data. For example, augmented datasets might shuffle a small subset of real-world data captures into a larger synthetic dataset [12], inject synthetic waveforms into captured spectrum, overlay multiple captured observations to create a more congested
Figure 2.2: An overview of the traditional ML and TL training pipeline.

observation [56], or add synthetic noise to real world captures [22, 57]. Augmented datasets are useful in testing detection and classification performance of signals in a congested or interference-heavy environment, but do not yield the highest performance per observation, when compared to captured data [12]. Further, there are a multitude of open research questions related to the development and use of augmented datasets. The examination of TL performance using augmented datasets is left for future work.

2.4 Common Approaches & Neural Network Architectures

RFML approaches have generally used DL techniques, a subset of ML techniques, over traditional ML algorithms. Traditional ML algorithms (i.e. linear and logistic regression, SVMs, and decision trees) are typically heavily impacted by the choice of data representation and do not functionally operate on raw RF data [58]. In contrast, DL approaches automa-
cally extract features from data using consecutive non-linear transformations [59], and are therefore better suited to learning from raw RF data.

Additionally, the vast majority of RFML works use supervised learning techniques, which require a label (or labels, in the case of multi-task learning) associated with each example within the training, validation, and test datasets [58]. Unsupervised/self-supervised and semi-supervised learning techniques are less common in the RFML space, but have been used in the channel modeling/emulation and spectrum anomaly detection works discussed previously in Section 2.2, as well as for signal representation learning [60]. It is also worth noting that in the context of wireless communications research, reinforcement learning has typically relied upon expert-defined features and/or measurement data [61], and therefore is not within the scope of this dissertation, but has shown success in applications such as radio control [62] and resource management [63].

Like in other DL applications, in RFML settings, a NN, $f$, aims to approximate some function $f^*$, such that $y = f(x, \theta)$, where $x$ is the network input (i.e. raw RF data), $y$ is the network output (i.e. modulation class, Tx ID), and $\theta$ is the parameters of the network. During training (Figure 2.2), an optimization procedure such as Stochastic Gradient Descent (SGD) [64], Adagrad [65], or Adam [66], is used to learn the value of $\theta$ that achieves the best approximation of $f^*$, as measured by a loss/cost function such as Mean Squared Error (MSE), Negative Log-Likelihood (NLL), or Cross-Entropy [58]. In the RFML literature, $f$, most commonly takes the form of a CNN or RNN, as highlighted in Table 2.1. Each is described in further detail in the following subsections, as well as MLPs, to provide context. Further background on DL, including mathematical foundations, learning mechanisms, and practical design discussions, can be found in [58].
2.4.1 Multi-layer Perceptrons (MLPs)

The most basic NN architecture is the MLP, depicted in Figure 2.3. MLPs are fully connected NN; every neuron in layer $i$ is connected to every neuron in layers $i-1$ and $i+1$. MLPs are universal function approximators, and can therefore approximate any continuous function using a single hidden layer, but are inefficient at large widths and depths and are not translation invariant [67]. Therefore, while MLPs have successfully been applied to modulation classification using hand-crafted or statistical features as input [68], they are not well suited to ingesting raw RF data.

2.4.2 Convolutional Neural Networks (CNNs)

CNNs address the scaling and translation invariance issues of MLPs by using sparse local connectivity and parameter sharing. More specifically, in a CNN, the learned weights are used as a filter, convolved over localized regions of the data, as shown in Figure 2.5. As a result, CNNs are position invariant, scale invariant, and have smaller memory footprints.
2.4. **Common Approaches & Neural Network Architectures**

Figure 2.4: An CNN with example input.

Figure 2.5: The 1D convolution operation, with a stride of 1 and no padding.

when compared to MLPs [58]. CNNs are an intuitive fit for use in RFML, as the concept and application the convolutional operator are familiar to the Digital Signal Processing (DSP) space. Furthermore, each filter can be interpreted as the impulse response to a Finite Impulse Response (FIR) filter [69]. Position and scale invariance is also thought to translate to “invariance to linear mixing, rotation, time shifting, [and] scaling” in RFML models [70].

Most basic CNN architectures begin with several convolutional layers interleaved with regularizers such as Dropout [71] or Batch Normalization [72] as the feature extraction step, as shown in Figure 2.4. The convolutional layers are generally followed by a set of linear/fully-connected layers that act as the decision maker. Convolutional layers are present in all of the model architectures used in this dissertation which are discussed in further detail in Chapter 4.
2.4.3 Recurrent Neural Networks (RNNs)

While MLPs and CNNs are feed-forward networks, RNNs include feedback loops or cyclic connections in which the outputs of the network are fed back into the network for further processing [58]. While RNNs come in many flavors and can be as simple as that shown in Figure 2.6, the most effective forms of RNNs are those that implement strategies for learning across multiple time scales such as leaky units and gated RNNs (i.e. LSTMs or Gated Recurrent Units (GRUs)) [58]. RNNs are also well suited to RFML applications, as they are specifically designed to process sequential data [73] and model non-linear Infinite Impulse Response (IIR) filters [74]. Recurrent layers are used in one of the two model architectures chosen for this dissertation, the CLDNN, described in Chapter 4.

2.5 Challenges & Obstacles to Widespread Deployment

Research into the application of DL technologies to RFML is accelerating and has demonstrated particular success in improving and automating spectrum situational awareness applications and supporting the next-generation of Cognitive Radio (CR) and cellular commu-
2.5. Challenges & Obstacles to Widespread Deployment

However, a lack of works holistically look at all of the considerations for making these systems deployable in real-world applications. As a result, RFML lags significantly behind more mature deep learning technologies such as CV and NLP. The following subsections briefly highlight three key challenges and areas for future research that are amongst the most salient needs that must be addressed, namely robust confidence metrics in RFML-derived decisions, real-time processing improvements, and, finally, online and TL techniques, the topic of this dissertation.

2.5.1 Human-Machine Interaction & End-User Confidence

While DL technologies have shown the ability to solve complex and hard to model problems, both within RFML and other application spaces, the black-box nature of their decision making process hampers their widespread adoption [75]. In particular, while DL systems can provide decisions to the user, they typically do not provide a good justification or confidence in their decision to the end-user, limiting the utility of the system outputs.

Beyond trusting individual decisions, additional work is also needed to help the end-user understand the limits of the learned behaviors, how to shape and/or optimize the system, and how to visualize and/or verify whether the machine should be trusted. In the same vein, additional work is required to identify in real-time if the current inputs are representative of the training data, a topic discussed further in Section 8. Such methods are not only needed in order to provide assured performance, but to begin ruggedizing the decision chain against spoofing and other adversarial techniques [76].
2.5.2 Real-time Processing Capabilities

The widespread availability and adoption of Graphical Processing Units (GPUs) have vastly accelerated the research and deployment of DL-based image processing applications. While GPUs have certainly accelerated RFML-based applications as well, the sequential time-series nature of RF data may require novel hardware processing architectures to facilitate further acceleration of these data types. In particular, recent research has demonstrated the applicability of Field-Programmable Gate Array (FPGA)-based implementations that greatly accelerate sequential data streams and may prove fruitful for real-time RFML processing [77]. The ability to process RF data and make decisions on a sample-by-sample basis allows for quicker, more agile, decision making which is incredibly important for the RFML application spaces considered in this work [78].

2.5.3 Online & Transfer Learning Techniques

As discussed in Section 2.4, current RFML systems predominately utilize supervised learning solutions in which the training process is performed offline, before deployment; the learned model remains fixed during deployment. The inflexibility of these systems means that, while they are appropriate for the conditions assumed during offline training, they are largely not adaptable to changes in the propagation environment and Tx/Rx hardware. Given the fluidity of modern communication environments, this rigidness greatly limits widespread adoption of RFML solutions. Additionally, many RF systems offer the potential for multiple apertures/nodes whose spectrum observations can be integrated to gain a larger system picture.

Research and development is needed to allow for online learning and transferring learned behaviors between platforms, environments, and use-cases. Solutions must consider that the
behaviors learned at one node will be influenced by their RF hardware, which is distinct and possibly vastly different from a second node, and that any behaviors learned in one environment may be distinct from another. For example, an RFML model trained on data captured in an empty field will learn different distinguishing features than an RFML model trained on data captured in a city center, because these two RF environments impact the received data in very different ways. In order to overcome these differences, environment adaptation, a subset of domain adaptation, will be necessary.

In this dissertation, TL performance is examined across these changes in RF hardware and propagation environment, both synthetically modeled and observed in captured data, generalizing over use-case and NN architecture. Through performance analysis and the use of specialized TL and dataset similarity metrics, trends are observed which point to the underlying system parameters affecting TL behavior and offer general guidelines and best practices for using RF TL in deployed systems.
Chapter 3

Transfer Learning for RFML

Although a general TL taxonomy exists, as well as Reinforcement Learning (RL), CV, and NLP specific taxonomies, these fail to consider challenges unique to the RF modality regarding the significant impacts of the RF platform and channel environments on learned behavior. This chapter presents a new domain-specific TL taxonomy for RFML and surveys the small body of existing works utilizing TL in the context of RFML, including the methods used and patterns observed in the results across these existing works. Section 3.1 provides the requisite notation and definitions used in the remainder of the work, background for why TL is necessary to accelerate research in RFML, and examples of how the terms domain and task can be interpreted in an RFML context. In Section 3.2, the general TL taxonomy from which our RFML-specific taxonomy is derived is briefly discussed. Section 3.3 presents the RFML-specific TL taxonomy, discusses specific motivations for RFML TL applications, and highlights the adaptations made to the general taxonomy introduced previously. Additionally, Section 3.3 surveys the small body of existing works in TL for RFML, discuss the methods used to achieve transfer in these works, and highlight initial trends seen across the works. Finally, Section 3.4 concludes the chapter.

This chapter is an adaptation of the following publication:

3.1 Definitions

As commonly accepted in the literature, this work uses the TL notation introduced in [79]: a domain $D = \{X, P(X)\}$ is comprised of the input data $X$ and the marginal probability distribution over the data $P(X)$, where $X = \{x_1, \ldots, x_n\} \in \mathcal{X}$, where $\mathcal{X}$ denotes the input space. The task $T = \{Y, P(Y|X)\}$ is comprised of the label space $Y$, and the conditional probability distribution $P(Y|X)$ learned from the training data pairs $\{x_i, y_i\}$ such that $x_i \in X$ and $y_i \in Y$. Generally, for RFML, the domain consists of the RF hardware and the channel environment, and the task comprises the application being addressed, including the range of possible outputs. Example elements of RFML domains and tasks are given in Table 3.1.

The source domain and task, denoted $D_S = \{X_S, P(X_S)\}$ and $T_S = \{Y_S, P(Y_S|X_S)\}$, are those defined during the initial training process. That is, the source domain and task describe the initial training data and labels. The target domain and task are denoted $D_T = \{X_T, P(X_T)\}$ and $T_T = \{Y_T, P(Y_T|X_T)\}$, and describe the intended use-case of the trained ML model. Note that labeled data may or may not be available for the target domain and task, or may only be available in limited quantities.

Traditional supervised ML techniques assume that $D_S = D_T$ and $T_S = T_T$ [79], allowing direct transfer to be employed with success. That is, the model trained for the source domain and task can be used for the target domain and task with no modification. However, in the context of RFML, inherent hardware variations and channel effects all but guarantee that $D_S \neq D_T$, unlike in the fields of CV and NLP. TL is motivated by this mismatch between the source and target domains and/or tasks, inhibiting direct transfer. More specifically, the aim of TL is to leverage the knowledge $P(Y_S|X_S)$ obtained using $D_S$ and $T_S$ to improve the performance of $P(Y_T|X_T)$ on $D_T$ and $T_T$ [79].
CHAPTER 3. TRANSFER LEARNING FOR RFML

Domain Elements | Tasks
--- | ---
• SNR | • \(n\)-class AMC
• AWGN | • SEI
• Ricean Fading | • Localization
• Multipath Effects | • Signal Detection
• Doppler | • End-to-End Communications
• Bandwidth | • SNR Estimation
• Sample Rate | • IQ Imbalance Estimation
• Noise Floor | • Signal Compression
• IQ Imbalance |  
• Phase Imbalance |  
• Non-linear distortion |  

Table 3.1: Example RFML Domain Elements and Tasks.

TL is feasible because a model trained on a source domain and task has learned generic knowledge about the structure of raw RF signals through the source domain/task, which may be used as prior knowledge to solve the target task. The utility of these previously learned features for solving the target task is dependent on the “similarity” between the source and target tasks and domains. On the one hand, the learned features used to perform AMC on a low cost IoT Tx/Rx are likely quite similar to those used to perform AMC on a high cost military-grade transceiver. On the other hand, the learned features used to perform AMC are likely different from those learned when performing SEI. The “similarity” between the source and target domains/tasks is not well-defined, a topic further discussed in Chapter 7, but can be though of as a continuous two-dimensional spectrum, depicted in Fig. 3.1, and dictates the success of TL.

To frame this discussion, consider the following four scenarios and examples in which \(D_S \neq D_T\) and/or \(T_S \neq T_T\) can occur:

1. \(P(Y_S|X_S) \neq P(Y_T|X_T)\) - The source and target tasks have different conditional probability distributions. This most commonly manifests in the form of unbalanced datasets, where a subset of classes have more examples in the source dataset than the target
### Table 3.2: The settings which describe points (a)-(i) on the two-dimensional spectrum of “similarity” between source and target domains and tasks shown in Fig. 3.1.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>The traditional ML setting where the source and target domains and tasks are the same.</td>
</tr>
<tr>
<td>(b)</td>
<td>The TL setting in which learned features from one domain are used to support performing the same task in a second domain. For example, using features learned to perform AMC in an AWGN channel to support performing AMC in a fading channel.</td>
</tr>
<tr>
<td>(c)</td>
<td>The setting in which the source and target domains are so dissimilar that TL is unsuccessful, despite the source and target tasks being the same.</td>
</tr>
<tr>
<td>(d)</td>
<td>The TL setting in which learned features from one task are used to support a second task, while the source and target domains are the same. For example, using features learned to perform AMC to support SEI with the source and target domains being the same.</td>
</tr>
<tr>
<td>(e)</td>
<td>Likely the most challenging TL setting in which learned features from one domain and task are used to support performing a second task in a new domain. For example, using features learned to perform AMC in an AWGN channel to support performing SEI in a fading channel.</td>
</tr>
<tr>
<td>(f)</td>
<td>The setting in which the source and target domains are so dissimilar that TL is unsuccessful, although the source and target tasks are somewhat similar.</td>
</tr>
<tr>
<td>(g)</td>
<td>The setting in which the source and target tasks are so dissimilar that TL is unsuccessful, despite the source and target domains being the same.</td>
</tr>
<tr>
<td>(h)</td>
<td>The setting in which the source and target tasks are so dissimilar that TL is unsuccessful, despite the source and target domains being somewhat similar.</td>
</tr>
<tr>
<td>(i)</td>
<td>The setting in which both the source and target tasks and domains are dissimilar, preventing the use of successful TL.</td>
</tr>
</tbody>
</table>
dataset or vice versa. A simple example might be transferring an AMC model between two datasets, both of which only contain BPSK and QPSK signal types. However, the source dataset contains 70% BPSK signals and 30% QPSK signals, while the target dataset contains 30% BPSK signals and 70% QPSK signals.

2. $Y_S \neq Y_T$ - The source and target tasks have different label spaces. For example, the target task contains an additional output class (i.e. for an AMC algorithm, the source task is a binary BPSK/QPSK output set, while the target task includes a third noise-only class). Alternatively, the target task may be completely unrelated and disjoint from the source task (i.e. the target task is to perform SEI while the source task was to perform AMC), therefore the label spaces are also disjoint.
3. $P(X_S) \neq P(X_T)$ - The source and target domains have different data distributions. An example of such a scenario includes transfer of models from one channel environment to another.

4. $X_S \neq X_T$ - The source and target feature spaces differ. An example includes performing SEI using the same set of known Txs, but using different modulation schemes in the source and target domain.

These scenarios are not mutually exclusive. That is, for any given TL setting, several of the above scenarios may be encountered. For example, when $Y_S \neq Y_T$ (Scenario 2) and/or $P(X_S) \neq P(X_T)$ (Scenario 3), the source and target tasks typically also have different conditional probability distributions (Scenario 1).

### 3.2 Other TL Taxonomies

Before presenting a TL taxonomy for RFML in the next section, we overview the general TL taxonomy presented in [79], from which our taxonomy builds. This taxonomy, or some adaptation thereof, is used in a number of fields including NLP [80] and CV [81]. While there are the number of ways to categorize TL problems [82], [79] categorizes the broad field of TL into three sub-fields – *unsupervised, inductive, and transductive* – each characterized by the availability of training data in the source and/or target domains, and whether or not the source and target tasks differ.

In *unsupervised TL*, no labeled data is available in either the source and target domains. The source and target tasks can be the same or different. *Inductive TL* settings are characterized by the availability of labeled data in the target domain, when the source and target tasks differ. Labeled data may or may not be available in the source domain. *Inductive TL* is
further broken out into:

- *self-taught learning* methods which address settings where no labeled data is available in the source domain,

- *multi-task learning* which assumes the availability of labeled data in both the source and target domains and in which the source and target tasks are learned simultaneously, and

- *sequential learning* which also assumes the availability of labeled data in both the source and target domains, however, the source task/domain is learned first and the target task/domain is learned second.

It should be noted that *sequential learning* was not included in the taxonomy presented by [79], but was detailed in [80], as it is an oft-utilized approach in the DL literature and a critical component of the meta-learning, life-long learning, and representation learning fields.

In *transductive TL* settings, no labeled data is available in the target domain, while the source and target tasks are the same. *Transductive TL* is further broken out into:

- *domain adaptation*, under which the source and target domains differ, and

- *sample selection bias*, also known as *covariance shift*, which refers to when both the source and target domains and tasks are the same, but the source and/or target training dataset may be incomplete or small.

### 3.3 An RFML-Specific Taxonomy & Existing Work

The proposed TL taxonomy for RFML is shown in Fig. 3.2, and is adapted from the general taxonomy discussed previously to contain the TL contexts most relevant to the current state-
### 3.3. AN RFML-SPECIFIC TAXONOMY & EXISTING WORK

<table>
<thead>
<tr>
<th><strong>TL Setting</strong></th>
<th><strong>Use Case</strong></th>
<th><strong>Source Domain</strong></th>
<th><strong>Source Task</strong></th>
<th><strong>Target Domain</strong></th>
<th><strong>Target Task</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment Adaptation</td>
<td>Move a Tx/Rx pair equipped with an AMC model from an empty field to a city center</td>
<td>Single Tx/Rx pair, AWGN channel</td>
<td>Binary AMC (BPSK/QPSK)</td>
<td>Same Tx/Rx pair, Multi-path channel</td>
<td>Binary AMC (BPSK/QPSK)</td>
</tr>
<tr>
<td>Platform Adaptation</td>
<td>Transfer an AMC model between UAVs</td>
<td>Single Rx, Many Tx, Fading channel w/ Doppler</td>
<td>Binary AMC (BPSK/QPSK)</td>
<td>Different Rx, Same Tx set, Fading channel w/ Doppler</td>
<td>Binary AMC (BPSK/QPSK)</td>
</tr>
<tr>
<td>Environment Platform Co-Adaptation</td>
<td>Transfer an AMC model between a ground-station and a UAV</td>
<td>Single Rx, Many Tx, Multi-path channel</td>
<td>Binary AMC (BPSK/QPSK)</td>
<td>Different Rx, Same Tx set, Fading channel w/ Doppler</td>
<td>Binary AMC (BPSK/QPSK)</td>
</tr>
<tr>
<td>Multitask Learning</td>
<td>Simultaneous signal detection and AMC</td>
<td>Single Tx/Rx pair, AWGN channel</td>
<td>Binary AMC (BPSK/QPSK)</td>
<td>Same Tx/Rx pair, AWGN channel</td>
<td>SNR Estimation</td>
</tr>
<tr>
<td>Sequential Learning</td>
<td>Addition of an output class(es) to an AMC model</td>
<td>Single Tx/Rx pair, AWGN channel</td>
<td>Binary AMC (BPSK/QPSK)</td>
<td>Same Tx/Rx pair, AWGN channel</td>
<td>Four-class AMC (BPSK/QPSK/16QAM/64QAM)</td>
</tr>
</tbody>
</table>

Table 3.3: Representative examples for TL settings in RFML.
of-the-art RFML algorithms. More specifically, given the limited use of unsupervised and self-supervised algorithms in the RFML literature, this taxonomy assumes the availability of some labeled data in both the source and target domains, though the size of these labeled datasets may be limited. This restricts the discussion herein to inductive TL techniques. However, as the field of RFML grows to encompass unsupervised and self-supervised techniques, this taxonomy can easily be expanded to include transductive learning techniques like those presented in the general taxonomy of [79].

In addition to limiting discussion to inductive TL techniques, three key changes have been made specific to the RF domain: first, while \textit{domain adaptation} is considered a transductive TL approach in the general taxonomy of [79], this taxonomy only considers domain adaptation approaches that make use of labeled source \textit{and} target data. Second, similar to the RL TL taxonomy presented in [83], the area of domain adaptation is further broken out into three categories: environment adaptation, platform adaptation, and environment platform co-adaptation. This alteration specifies the type domain change, and highlights that an
environmental shift (i.e. a channel change) is vastly different than a change in Tx or Rx hardware, as described further in the subsequent subsections. Third, like in [80], sequential learning is included in this taxonomy, to provide a counterpart to multi-task learning. For clarity, representative examples for each TL setting described given in Table 3.3, and are expounded upon in the following subsections with parallels drawn to other modalities where appropriate.

3.3.1 Domain Adaptation

When the source and target tasks are the same, but the source and target domains differ, domain adaptation is required in form of environment adaptation, platform adaptation, or environment platform co-adaptation. More specifically, domain adaptation techniques are needed when the label space remains constant (i.e. \( Y_S = Y_T \)) and the conditional probability distributions learned from the source and data sets is the same (i.e. \( P(Y_S|X_S) = P(Y_T|X_T) \)), but the source and target domains have different data distributions (i.e. \( P(X_S) \neq P(X_T) \)) and/or the source and target feature spaces differ (i.e. \( X_S \neq X_T \)). However, the cause of different source and target data distributions and/or feature spaces can be caused by either a change in platform (i.e. Tx and/or Rx hardware) or environment.

Environment Adaptation

In the context of RFML, the aim of environment adaptation is to adapt a learned model to a changing channel environment, while holding the Tx/Rx pair(s) constant. Environmental factors such as time of day, temperature, atmospheric conditions, channel type, and any movement of the Tx and/or Rx may potentially create variations in signal capture which has the potential to affect the learned behavior of an RFML system. Consider the representative
example of moving a Tx/Rx pair equipped with an AMC model from an empty field to a city center. Though the Tx/Rx pair stays constant, the channel shifts from a line-of-sight, likely AWGN, channel to an environment with significant multi-path effects and interference from neighboring devices. Such an example is similar to performing image classification indoors versus outdoors [84], or utilizing image classification algorithms in environments where the captured image may degraded by weather conditions [85].

Few works have examined the impact of a changing environment on RFML performance, and as a result, little is known about the extent to which the parameters given above may prevent transfer between environments. However, existing work used fine-tuning techniques to successfully transfer RFML models from one real environment to additional real environments [86], or between two synthetically modeled channel environments [87]. More specifically, in [86], a robust DL-based spectrum sensing framework was proposed which used techniques similar to those used for sequential learning to adapt pre-trained models to changing wireless conditions with little-to-no labeled target data. A similar approach was used in [87] to adapt a model pre-trained on the popular RML2016.10a dataset [88] to a custom synthetically generated dataset with the same modulation types for an AMC use case.

If works using spectrograms as input to a DL model are considered, rather than raw RF data, [89] presents a CNN-based SVM approach to perform non-cooperative spectrum sensing. In this work, an AlexNet inspired CNN was used as a naive feature extractor, and a linear SVM was used to determine whether or not the spectrum band-of-interest was occupied using the features extracted by the CNN. When the environment or location changed, the initial layers of the CNN feature extractor were frozen, while the remaining layers and the SVM were retrained using data from the new environment or location. Results showed that TL reduced the number of spectrograms needed to achieve the same performance without TL most significantly when transferring from environments with low SNR levels to environ-
ments high SNR levels. Some performance improvements were also seen when transferring between environments with similar SNR levels. However, performance typically degraded when transferring from high SNR levels to low SNR levels, a phenomenon known as negative transfer [90, 91].

**Platform Adaptation**

In contrast to environment adaptation, the aim of platform adaptation is to overcome changes in Tx/Rx hardware while holding the channel environment constant. Variations in hardware non-linearities, IQ imbalances, or frequency, phase, and/or timing offsets all have the potential to inhibit model transfer between platforms. Additionally, note that while the Rx hardware will always be user-controlled, the Tx may or may not be. That is, changes in Tx hardware will be outside the control of a third party listener, further complicating the task.

A representative example of platform adaptation includes transferring an AMC model between Unmanned Aerial Vehicles (UAVs). Presuming the UAVs are flying in the same vicinity, the channels they encounter will be similar. Additionally, the received signals on both platforms will be effected by Doppler shifts. However, small hardware variations caused by manufacturing inconsistencies, age, settings, etc. will cause variations between the signals received on each platform. While one might draw parallels between transferring models between RF platforms and between cameras capturing images for CV algorithms, it seems that assuming the two cameras are capable of capturing images at the same resolution, such a transfer does not affect performance in any significant way [13]. However, [9] showed that directly transferring learned models between transmitter and receiver pairs diminished performance by as much as 7%, even when augmentations, such as adding noise, FOs, and resampling, were applied to the training data. These results have been echoed in several subsequent works, both in the context of AMC [12] and SEI [92], which have aimed to mitigate
performance degradation through data augmentation/transformation, data preprocessing, or training over a variety of platforms. Yet, despite the growing body of work recognizing the need for platform adaptation methods, little work has been done to identify the impact of changing hardware platforms on RFML performance, or to develop methods for transfer between hardware platforms.

**Environment Platform Co-adaptation**

Finally, environment platform co-adaptation combines the challenges of environment adaptation and platform adaptation with the goal of transferring a learned model to a new channel environment, as well as to a new Tx/Rx pair(s). It should be noted that transfer between data types (i.e. between synthetic, augmented, or captured datasets) is also considered environment platform co-adaptation, as modeled hardware effects and channel conditions will inherently be different from those found in captured data (Section 2.3).

As a representative example of environment platform co-adaptation, consider transferring an AMC model between an RFML-enabled ground-station and a UAV. In such a scenario, not only will the change in hardware impact the received signals and resultant performance, but the channel environments encountered by the two devices will differ significantly. Because changes in CV platform (i.e. cameras) do not impact performance in the same way that changes to the RF platform does, as discussed above, environment platform co-adaptation is a scenario not typically discussed in the CV literature, but is akin to techniques aimed at domain adaptation using drawings or clip art as a source domain and real images as the target domain [93]. Conceptually, this is similar to transferring models between synthetic and captured data in the RF space.

In the context of RFML, existing works in the area of environment platform co-adaptation
have primarily taken the form of DL models pre-trained on synthetic data and fine-tuned using captured data. More specifically, [16] and [94] examine transferring residual and Autoencoder (AE)-based models from synthetic to real environments for AMC and channel model estimation problems. These works fine-tune varying amounts of the pre-trained NN—just the final layer in [16] and the latter half of the NN in [94]—for a small number of epochs with a small learning rate, taking cues from the CV literature [95]. If the use of power spectrums are considered as input, [2] has also examined the transfer of CNNs from synthetic to real environments for signal detection. In this work, the entire CNN is tuned, again for a small number of epochs with a small learning rate.

Work in [96] uses a TL-based approach to adapt pre-trained RF fingerprinting models to each edge node in a federated learning scenario. While the authors only acknowledge the intent to transfer across channel environments, in practice, this approach could and would also be used to mitigate the impacts of the Rx hardware on the performance of the model at each edge node. Regardless, the addition of TL to the federated learning algorithm significantly increased in performance when compared to baseline federated and centralized learning approaches, and performance was maintained across long-term changes in the channel environment through iterative model updates, tested using real-world data.

Recent work has also confirmed the intuitive result that the order in which datasets are trained is critical to achieving successful transfer of learned behaviors [97]. More specifically, in [97], the authors showed that when comparing the performance of an AMC model trained on synthethic data, augmented data, captured data, or some combination thereof, the best performance on captured test sets (which are representative of what will be observed once deployed) is achieved when pre-training on synthetic datasets and using the captured data for fine-tuning. That is, synthetic and augmented data are best for pre-training, while the captured data, which is typically smaller in quantity anyhow, is best for fine-tuning.
Such results do make the realistic assumption that the final trained model will be evaluated/tested/deployed on real captured data, the implications of which are discussed further as a part of *sequential learning* in Section 3.3.2.

### 3.3.2 Sequential Learning

Finally, sequential learning describes the setting in which a source task is learned first, and the aim is to transfer the pre-trained model to a different target task, typically via fine-tuning techniques \[^{95}\] , like those used for domain adaptation. In ever-changing wireless conditions, sequential learning will be a critical component of future online, lifelong, and meta-learning techniques for RFML systems. For example, adding output class(es) to a pre-trained AMC model can be considered a representative example of sequential learning. Such an approach was examined in \[^{26}\] , wherein sequential learning techniques were used to fine-tune a pre-trained residual CNN for 190 entirely new categories/output classes, with as few as 50 to 500 samples per category, versus a model trained from random initialization. (The pre-trained model had been trained for using 13,000,000 training samples from over 5,000 categories.) Results showed that the fine-tuned models not only converged over an order of magnitude faster than the model trained from random initialization, but achieved higher test accuracies as well. A similar approach is used in Chapter 6 to perform successive refinement of models by adding/removing a single signal type to the task at a time in Chapter 6.

Additional work in \[^{98}\] examined the use of sequential learning methods for adapting pre-trained SEI models for intended use cases, including tuning for changes in Txs (i.e. output classes) and protocols used. More specifically, this work built upon an existing architecture, RiftNet \[^{46}\] , using supervised pre-training and fine-tuning methods similar to those discussed above, as well as unsupervised pre-training and TL methods, such as the use of reconstruction
losses and manifold clustering, for novel device detection/classification. This work further examined the impact of source/target dataset size, the number of source/target output classes, and changes in protocol between the source and target on transferability. When using supervised pre-training and fine-tuning methods, results showed that pre-training on larger, more diverse source datasets provided the best TL result, hypothesizing that such models learned the most generalizable features for the domain. The fine-tuning of these pre-trained models out-performed the baseline classifiers trained from random initialization in most all cases, and performance was best when only the relevant output classes were retained and extraneous output classes were removed. As expected, the larger source and target datasets yielded higher performance, with the size of the source dataset having a slightly larger impact on end performance than the size of the target dataset. However, overfitting was common during the fine-tuning process, requiring care and attention in the setting of hyperparameters and use of early stopping and/or checkpoint methods. Additionally, transfer was more challenging between protocols, requiring additional fine-tuning steps and resulting in low top-1 accuracy. When using unsupervised reconstruction-based TL methods, results showed that the use of multi-burst processing, batching 5 signals from the same Tx together, provided additional context for the model during the reconstruction process that yielded the best performance. Further, the reconstruction-based TL methods were more capable of overcoming differences in protocol between the source and target dataset.

Universal representation learning, a pre-training method, has also been explored in the context of RFML, but has yet to become as ubiquitous in the CV and NLP fields [60, 98, 99, 100]. Universal representation learning approaches aim to learn general purpose features or embeddings which “capture the generic factors of variation present in all the classes” that can be used between tasks [101]. Such approaches are then used as feature extractors or fine-tuned for the target task(s). That is, universal representation learning is a source task that
aims to provide successful transfer to a variety of target tasks, significantly decreasing the training time for downstream algorithms.

3.3.3 Multi-task Learning

The aim of multi-task learning is to learn differing source and target tasks simultaneously, and is typically characterized by the use of more than one loss term during training. This encourages the model to learn more general features that are useful in multiple settings. For example, an RFML model trained to simultaneously perform signal detection and AMC will likely learn more general features about signal structure and modulation than an RFML model trained to perform only one of these tasks.

Multi-task learning has perhaps been explored more frequently than any other TL setting in the context of ML-enabled wireless communications using expert-defined features. For example, in [102], a multi-task learning architecture is designed for joint jamming detection/localization and link scheduling, in [103] and [104], CNN and RNN architectures are trained to perform Wi-Fi and cellular traffic forecasting, predicting the maximum, minimum, and average load and the load across neighboring cells, respectively, and work in [105] used CNNs to perform indoor Wi-Fi localization.

However, examples of multi-task learning in the context of RFML, as defined in this work, are far fewer. The limited body of works include an approach for end-to-end communications presented in [106], as well as several which have explored multi-task learning as a way to both improve the explainability and accuracy of models trained to perform Automatic Modulation Classification (AMC). More specifically, in both [18] and [107], modulation classes are broken into subgroups, either by modulation type (i.e. linear, frequency, etc) or in order to separate the modulation schemes which cause the most confusion (i.e. 16QAM
and 64QAM), and in [108], concept bottleneck models were used to provide inherent decision explanations while performing AMC via the prediction of a set of intermediate concepts defined prior to training. Finally, [109] presents an approach for simultaneous AMC and SEI using a DenseNet and Transformer-based feature extraction step, followed by a mask-based dual-head classifier, trained using a multitask cross-entropy loss. This approach yields near state-of-the-art performance on both tasks, but was only tested on a custom synthetic dataset.

3.4 Discussion

Every area of TL in the context of RFML remains an open area of research. While algorithmic development is the most apparent direction for future work, the gap in understanding how the channel environment and platform variation impact transfer learning behavior limits the ability to design TL algorithms for RFML. The survey of works in Section 3.3 has highlighted that borrowing methods used in other modalities can yield success. However, such limitations in understanding may hinder further performance improvements that might be yielded from TL techniques and also obscure insights into long-term model behavior during deployment, which has long been a criticism of RFML and prevented commercial support and deployment [7]. In the next chapter, we will describe the experimental framework designed to explore how environmental factors and hardware considerations impact learned behavior and may therefore encourage or prevent transfer, reducing this knowledge gap.
Chapter 4

An RF TL Experimental Framework

In an effort to generalize over use-case and NN architecture when analyzing TL performance, the experiments performed in this dissertation span two unique use-cases, Automatic Modulation Classification (AMC) and Specific Emitter Identification (SEI), and two NN architectures, a small and simple Convolutional Neural Network (CNN), as well as a Convolutional Long Short-term Deep Neural Network (CLDNN). This chapter presents the experimental framework used throughout the remainder of this dissertation, beginning with a description of the synthetic and captured datasets created for this work in Section 4.1. While Section 2.2 described AMC and SEI in detail, Section 4.2 describes the architectures chosen to perform these tasks, and Section 4.3 describes the model pre-training, TL, and evaluation pipelines. In Section 4.4, several real-world considerations are discussed, including the impacts of different Tx hardware, Rx hardware, CFs, and channels on baseline model performance, how much performance degrades from domain to domain without TL, as well as the impacts of removing the CFO from the captured data on baseline and TL performance. Section 4.5 concludes the chapter.

This chapter is adapted from portions of the following publications:


4.1 Data

As previously discussed in Section 2.3, synthetic data is well-suited to initial development, as the data is quick and straightforward to generate and label, and allows for complete control over the parameters-of-interest. However, captured data is all but required for test and evaluation prior to real-world deployment, and includes degradations that can be difficult to model in synthetic data. As such, the first set of experiments performed herein uses synthetically generated data, and the second set of experiments builds off of the results of the first using captured data. More specifically, this work required a well-labeled synthetic dataset with a wide range of signal types, SNRs, and FOs, as well as a well-labeled captured dataset with multiple signal types, channel types, Tx IDs, and Rx IDs. The few publicly available synthetic and captured RF datasets contain known generation, collection, and/or labeling errors [88] or simply lack the signal types or parameters to be used in a study such as this [110]. To meet the needs of this dissertation, both a custom signal generation tool and a custom collection platform were built for this work, and are described below. Additionally, the synthetic RF dataset generation tool is open-source and available at https://github.com/IntelLabs/Synthetic-Radio-Frequency-Data-Generator.
4.1.1 Synthetic Data Generation

The synthetically generated data used in this dissertation was created using Python wrappers around the open-source signal processing library *liquid-dsp* \(^{[53]}\), which allowed for full control over the chosen parameters-of-interest, SNR, FO, and modulation type, and ensured accurate labelling of the training, validation, and test data. The data was generated using the same noise generation, signal parameters, and signal types as in \(^{[18]}\). More specifically, the signal space has been restricted to the 23 signal types shown in Table 4.1, observed at complex baseband in the form of discrete time-series signals, \(s[t]\), where

\[
s[t] = \alpha_\Delta[t] \cdot \alpha[t] e^{(j\omega[t] + j\theta[t])} \cdot e^{(j\omega_\Delta[t] + j\theta_\Delta[t])} + \nu[t]
\]  

(4.1)

where \(\alpha[t]\), \(\omega[t]\), and \(\theta[t]\) are the magnitude, frequency, and phase of the signal at time \(t\), respectively, and \(\nu[t]\) is the additive noise interference from the channel. Any values subscripted with a \(\Delta\) represent imperfections/offsets caused by the Tx/Rx and/or synchronization. Without loss of generality, all offsets caused by hardware imperfections or lack of synchronization have been consolidated onto the Tx during simulation.

Signals were initially synthesized in an AWGN channel environment with unit channel gain and no phase offset. The FO is not time-varying, and held constant for each observation.
### 4.1. DATA

<table>
<thead>
<tr>
<th>Modulation Name</th>
<th>Parameter Space</th>
<th>Modulation Name</th>
<th>Parameter Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPSK</td>
<td>Symbol Order {2}</td>
<td>APSK32</td>
<td>Symbol Order {32}</td>
</tr>
<tr>
<td></td>
<td>RRC Pulse Shape</td>
<td></td>
<td>RRC Pulse Shape</td>
</tr>
<tr>
<td></td>
<td>Excess Bandwidth {0.35, 0.5}</td>
<td></td>
<td>Excess Bandwidth {0.35, 0.5}</td>
</tr>
<tr>
<td></td>
<td>Symbol Overlap ∈ [3, 5]</td>
<td></td>
<td>Symbol Overlap ∈ [3, 5]</td>
</tr>
<tr>
<td>QPSK</td>
<td>Symbol Order {4}</td>
<td>FSK5k</td>
<td>Carrier Spacing {5kHz}</td>
</tr>
<tr>
<td></td>
<td>RRC Pulse Shape</td>
<td></td>
<td>Rect Phase Shape</td>
</tr>
<tr>
<td></td>
<td>Excess Bandwidth {0.35, 0.5}</td>
<td></td>
<td>Symbol Overlap {1}</td>
</tr>
<tr>
<td></td>
<td>Symbol Overlap ∈ [3, 5]</td>
<td>FSK75k</td>
<td>Carrier Spacing {75kHz}</td>
</tr>
<tr>
<td>PSK8</td>
<td>Symbol Order {8}</td>
<td></td>
<td>Rect Phase Shape</td>
</tr>
<tr>
<td></td>
<td>RRC Pulse Shape</td>
<td></td>
<td>Symbol Overlap {1}</td>
</tr>
<tr>
<td></td>
<td>Excess Bandwidth {0.35, 0.5}</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Symbol Overlap ∈ [3, 5]</td>
<td>GFSK5k</td>
<td>Carrier Spacing {5kHz}</td>
</tr>
<tr>
<td>PSK16</td>
<td>Symbol Order {16}</td>
<td></td>
<td>Gaussian Phase Shape</td>
</tr>
<tr>
<td></td>
<td>RRC Pulse Shape</td>
<td></td>
<td>Symbol Overlap {2, 3, 4}</td>
</tr>
<tr>
<td></td>
<td>Excess Bandwidth {0.35, 0.5}</td>
<td></td>
<td>Beta ∈ [0.3, 0.5]</td>
</tr>
<tr>
<td></td>
<td>Symbol Overlap ∈ [3, 5]</td>
<td>GFSK75k</td>
<td>Carrier Spacing {75kHz}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Gaussian Phase Shape</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Symbol Overlap {2, 3, 4}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Beta ∈ [0.3, 0.5]</td>
</tr>
<tr>
<td>OQPSK</td>
<td>Symbol Order {4}</td>
<td>MSK</td>
<td>Carrier Spacing {2.5kHz}</td>
</tr>
<tr>
<td></td>
<td>RRC Pulse Shape</td>
<td></td>
<td>Rect Phase Shape</td>
</tr>
<tr>
<td></td>
<td>Excess Bandwidth {0.35, 0.5}</td>
<td></td>
<td>Symbol Overlap {1}</td>
</tr>
<tr>
<td></td>
<td>Symbol Overlap ∈ [3, 5]</td>
<td>GMSK</td>
<td>Carrier Spacing {2.5kHz}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Gaussian Phase Shape</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Symbol Overlap {2, 3, 4}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Beta ∈ [0.3, 0.5]</td>
</tr>
<tr>
<td>QAM16</td>
<td>Symbol Order {16}</td>
<td>FM-NB</td>
<td>Modulation Index ∈ [0.05, 0.4]</td>
</tr>
<tr>
<td></td>
<td>RRC Pulse Shape</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Excess Bandwidth {0.35, 0.5}</td>
<td></td>
<td>FM-WB</td>
</tr>
<tr>
<td></td>
<td>Symbol Overlap ∈ [3, 5]</td>
<td></td>
<td>Modulation Index ∈ [0.825, 1.88]</td>
</tr>
<tr>
<td>QAM32</td>
<td>Symbol Order {32}</td>
<td>AM-DSB</td>
<td>Modulation Index ∈ [0.5, 0.9]</td>
</tr>
<tr>
<td></td>
<td>RRC Pulse Shape</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Excess Bandwidth {0.35, 0.5}</td>
<td></td>
<td>AM-DSBSC</td>
</tr>
<tr>
<td></td>
<td>Symbol Overlap ∈ [3, 5]</td>
<td></td>
<td>Modulation Index ∈ [0.5, 0.9]</td>
</tr>
<tr>
<td>QAM64</td>
<td>Symbol Order {64}</td>
<td>AM-LSB</td>
<td>Modulation Index ∈ [0.5, 0.9]</td>
</tr>
<tr>
<td></td>
<td>RRC Pulse Shape</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Excess Bandwidth {0.35, 0.5}</td>
<td></td>
<td>AM-USB</td>
</tr>
<tr>
<td></td>
<td>Symbol Overlap ∈ [3, 5]</td>
<td></td>
<td>Modulation Index ∈ [0.5, 0.9]</td>
</tr>
<tr>
<td>APSK16</td>
<td>Symbol Order {16}</td>
<td>AWGN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RRC Pulse Shape</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Excess Bandwidth {0.35, 0.5}</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Symbol Overlap ∈ [3, 5]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: The generation parameters and signal types included in the synthetic datasets.
Like in [18], SNR is defined as

$$\text{SNR} = 10 \log_{10} \left( \frac{\sum_{t=0}^{N-1} |s[t] - v[t]|^2}{\sum_{t=0}^{N-1} |v[t]|^2} \right)$$  \hspace{1cm} (4.2)$$

where $N$ is the length of the capture measured in samples. This definition of SNR is based on an oracle-style knowledge of the generated signals, where the symbol energy ($E_s$) has been calibrated relative to its instantaneous noise floor ($N_0$), with the sampling bandwidth being marginally higher than the actual signal bandwidth. It should be noted that RFML approaches generally ingest more than one symbol at a time increasing the effective SNR. Therefore, feature estimation and/or classification is supported at lower SNRs.

A blind receiver is assumed, so no synchronization or demodulation takes place. As a result, the conclusions of the experiments performed herein are not limited by any specific filtering approaches, bandwidths, or other baseband processing. All captures containing only AWGN have a Nyquist rate of 1, and the remaining captures have a Nyquist rate of either 0.5 or 0.33 (twice or three times the Nyquist bandwidth). While all signals are sampled at a sufficiently high rate to meet Nyquist’s sampling theorem, the DL and TL approaches used herein do not rely on this critical sampling assumption, as there is no attempt to reconstruct the original signal.

### 4.1.2 Captured Data Collection

The collection of a captured dataset suitable for this work required developing a custom collection platform. The Blind User-Reconfigurable Platform (BURP), shown in Figures 4.2 and 4.3, is a semi-portable hardware testbed that provides the means of creating robust datasets for up to 100 transmitting devices over a long period of time and in a variety of environments. More specifically, BURP coordinates a large number of Txs and multiple
Figure 4.2: Front bank of USB hubs on the Tx host machine with 20 Yard Stick One devices distributed throughout.

Figure 4.3: Side profile of the Tx host machine with the side panel removed.
Figure 4.4: The organization structure of data creation and synchronization of transmissions.

Rxs to automatically transmit, collect, and label data. The design of BURP emphasizes automation for long-term captures with remote monitoring, automatic error checking, and recovery procedures, as well as correct and complete labelling.

The proof-of-concept design uses YARD Stick One (YS1) radios, based on a Texas Instruments CC1111 MCU: a low cost unit that can transmit on a range of CFs (300-348 MHz, 391-464 MHz, 782-928 MHz) with different modulation schemes (ASK/OOK, GFSK, FSK2, FSK4, MSK), and at a range of power levels (-30 to +10 dBm) [111, 112]. The Rx end of the BURP platform hosts the actual data collection facilities, and utilizes multiple collection nodes. These collection nodes host USRPs, much higher resolution radios than the Tx host, and are portable enough to be easily moved, re-oriented, and re-configured to collect data with different channel conditions and multi-path effects. Each collection node handles its own data collection, tagging, and storage in coordination with the transmit end of the platform.

BURP transmissions are organized such that each task is a collection of runs, each run is a collection of frames, and each frame is a collection of bursts, as shown in Fig. 4.4. A burst is defined as a labeled time period with distinct start and end times where an emitter is transmitting. Each frame is a sequence of bursts with a transmitted header and trailer, used
for synchronization between the transmitting and receiving ends of the system. Each header and footer is a 300 baud FSK2 modulated sequence of 1023 ones, 1 zero, the unique frame ID, then 1 zero and 1023 ones. This, in effect, creates a pure-tone marker that can be used to cross-reference between Rxs the exact time when frames begin.

The operational flow of BURP, shown in Figure 4.5, begins and ends when prompted by the Tx end of the system through a control back-plane. The configuration for each run is specified in a JSON formatted file, located at the Tx end, specifying the sample space for the possible data rates, CFs, modulation schemes, burst payload sizes and contents, bursts per frame, and where to save data captures. First, the Tx host publishes that a run is beginning with information such as expected radios for collection, CF, sample rate, bandwidth, and gain. When this information is received, the collection nodes configure the attached Rx device(s) based on the provided parameters, begin collection, and return a ‘ready’ signal to
the Tx host. Once all collection nodes are ready, the Tx host then begins a ‘radio setup’
phase, where it logs all connected Tx IDs and performs a recovery sequence if any faults are
detected. The Tx host then generates a groundtruth sequence of Tx IDs and modulation
schemes for this run, and begins transmitting the first frame, starting with the frame header,
and noting the run and frame start-time in the groundtruth. Once all bursts in a frame are
complete, the trailer is transmitted to indicate the end of a frame. If any Txs become
unresponsive or stop working properly within a frame, the software will cycle power to the
faulty devices at the end of the frame in an effort to recover them. When all frames have
been transmitted, the current run ends and the next run begins. This cycle repeats until all
tasks are completed and the Tx software exits with a message to all collection nodes that
the task has ended. Throughout a task, the collection nodes will receive periodic updates
from the Tx host such as the timing of the beginning of a new run or frame, or changes in
collection parameters such CF or bandwidth. The collected data for each task is written
in one SigMF file per collection node, annotated with the back-plane-communicated timing
and metadata (i.e. detected Rx anomalies, and atmospheric conditions of temperature, dew
point, and pressure).

The post-processing of the captured data is split into two primary steps, as shown in Fig.
4.6: signal detection and burst isolation/labeling. Both are entirely implemented in Python.
The signal detection stage is performed blind using a simple energy detection algorithm
to identify the location (i.e. start/stop indices) of each burst within the provided SigMF
data file, not distinguishing between header, footer, and bursts. Detection is performed as
follows: after loading the complex IQ data from the SigMF data file and calculating the
magnitude of each sample, the magnitude array is iteratively smoothed. The magnitude of
the noise floor is estimated from the first $n$ samples of the smoothed magnitude, where $n$
is a tunable hyperparameter. Then, looping over the smoothed magnitude, when the value
of the smoothed magnitude is greater than the noise floor plus some threshold (a tunable parameter), the start of a burst is denoted. When the value of the smoothed magnitude returns below the noise floor plus threshold for at least $m$ samples (a tunable parameter), the end of the burst is denoted. Any extra or missed bursts detected are removed during the burst isolation and labeling stage of post-processing. Bursts with low SNR, such that the smoothed magnitude of the burst does not exceed the noise floor plus threshold, are missed and thus are not included in the dataset. Similarly, bursts with any sustained drops in magnitude, due to environmental factors, often trigger the detection algorithm multiple times and are also not included in the dataset, as the correct ground truth labels can be difficult to decipher.

Burst isolation and labelling uses the output of the previous signal detection stage, along with the SigMF data, metadata, and BURP groundtruth files to save each individual burst in separate SigMF data files with associated metadata containing the timestamp, estimated
SNR, CFO, modulation scheme, Tx/Rx ID, temperature, dew point, and atmospheric pressure measurements, and all other parameters denoted in the configuration file. First, header/footer bursts are identified using the groundtruth timestamps, and the location of each non-header/footer burst is verified by extrapolating from the header/footer timestamps using the sample rate. Any extra or missed bursts detected in the previous stage are identified through a mismatch in groundtruth timestamp and extrapolated time, and are removed from the detections and groundtruth lists. All headers and footers are also removed from the detections list, so as not to be included in the dataset. Then, for each verified burst in the detections list, the contents of the burst is saved to a SigMF data file with the name ‘runID_frameID_timestamp.sigmf-data’. All other available metadata (i.e. modulation scheme, Tx/Rx ID, temperature/humidity) is collected from the groundtruth files and saved in the associated ‘runID_frameID_timestamp.sigmf-meta’ file along with the estimated SNR of the burst, which is calculated using the noise floor before and after the burst, and the estimated CFO.

Additional details regarding the design of the experimental hardware setup, performance measurements, and support for potential future RFML experimentation can be found in [113].

### 4.1.3 Dataset Creation

For both the synthetic and captured datasets, large “master” datasets were created containing all modulation schemes and combinations of SNR and FO needed for the synthetic dataset experiments and all modulation schemes, Tx and Rx devices, propagation environments, and CFs needed for the captured dataset experiments. The synthetic master dataset is publicly available on IEEE DataPort [114], and the captured master dataset will also be
Figure 4.7: From each master dataset, subsets containing the desired metadata parameters are selected using configuration files.

publicly released at a later date. For each experiment performed, subsets of the data are selected from the either the synthetic or captured master dataset using configuration files containing the desired metadata parameters, as shown in Figure 4.7. Then, the relevant bursts are ingested, and split into segments of desired length within PyTorch. For example, only synthetically generated modulation schemes with SNRs between [0dB, 10dB] or bursts transmitted by device 37 and received by CN 2 can be selected. These bursts can then be split into examples of any length up to the total length of the burst. To ensure each training example is distinct from one another, a buffer is placed between each example taken from the data file, as shown in Fig. 4.8.

The synthetic master dataset contains 600000 examples of each the signal types given in Table 4.1, for a total of 13.8 million examples. For each example, the SNR is selected uniformly at random between [-10dB, 20dB], the FO is selected uniformly at random between [-10%, 10%] of the sample rate, and all further signal generation parameters relevant for the signal type, including symbol order, carrier spacing, modulation index, and filtering parameters (excess bandwidth, symbol overlap/filter delay, and/or beta), are selected uniformly at random
from the ranges specified in Table 4.1. Each example and the associated metadata is saved in SigMF format [115].

The captured master dataset includes transmissions from 30 YS1 emitters at three different CFs (346.3MHz, 416.4MHz, and 783.7MHz) and captured using three co-located collection nodes at each of three different Rx locations (in-room line-of-sight, in-room partial occlusion, and an adjoining room). Two of the three collection nodes hosted USRP B200s, while the third collection node hosted a USRP E310. Each run, three at each CF/Rx location combination, included 64 frames of 64 bursts, each of which had a payload of 1024 bytes of randomized data transmitted at 31,250 baud. The three co-located collection nodes were configured with a 250kHz sample rate and 250kHz bandwidth with a gain of 50dB. Each radio, Tx and Rx, was connected to an L-com 900 MHz 3 dBi rubber duck antenna which remained the same throughout the course of the collections. Between each run, the USB location of each transmitting device was randomized.

In total, the captured dataset contains 71.6 million examples of 128 IQ samples from 314435 bursts. While the number of examples from each collection node and CF is roughly equal, the dataset contains approximately twice as many examples from the partial occlusion and through wall locations as from the line-of-sight location. Because distance between the
4.2. MODEL ARCHITECTURES

Figure 4.9: The SNR of the examples in the captured dataset by capture location.

Txs and Rxs is significantly farther in the line-of-sight location compared to the partial occlusion and through wall locations, the SNR of the examples captured at this location are lower (Figure 4.9). Not only does this affect the detection scripts, but will also come into play when discussing domain performance and “difficulty” in Chapter 5. However, enough examples are present from each capture location to provide appropriately sized data-subsets for each domain and task examined in this work. In other words, the data-subsets created for each domain and task examined in this work contain the same numbers of examples per output class, regardless of the collection location.

4.2 Model Architectures

4.2.1 Convolutional Neural Network (CNN)

All AMC experiments performed using synthetic data utilize a single CNN architecture. Given the large number of models trained for these experiments (4698 total models), training
Figure 4.10: The CNN model architecture used for the synthetic AMC experiments, where \( n \) is the number of output classes (modulation types) trained.

time was a primary concern when selecting this model architecture. A larger, more complex architecture, described below, is used in the captured dataset experiments to show that the results of this work are model agnostic.

The simple CNN architecture chosen, shown in Figure 4.10, is based off of the architectures used in [18] and [108], with a reduction in the input size. Although many works including [18] and [108] have found success using larger input sequences, works such as [70] and [4] have found 128 input samples to be sufficient, especially when using synthetic data. Recognizing that longer input sequences results in increased computation and training time, in this work, 128 raw IQ samples are used as input corresponding to approximately 16-32 symbols depending on the symbol rate of the example. These samples are fed to the network in a \((1, 2, 128)\) tensor, such that 1 refers to the number of channels, 2 refers to the IQ components, and 128 refers to the number of samples. The network contains two 2D convolutional layers, the first uses 1500 kernels of size \((1, 7)\) and the second uses 260 kernels of size \((2, 7)\). The second convolutional layer is followed by a flattening layer, a dropout layer using a rate of 0.5, and two linear fully-connected layers containing 65 and \( n \) nodes where \( n \) is the number...
Figure 4.11: The CLDNN model architecture used for the captured AMC and SEI experiments, where \( n \) is the number of output classes (modulation types or emitter IDs) trained.

of output classes (i.e. modulation schemes) being trained. Both convolutional layers and the first linear layer use a ReLU activation function, and the final linear layer uses a Softmax activation function.

4.2.2 Convolutional Long Short-term Deep Neural Network (CLDNN)

All experiments performed using captured data (both AMC and SEI use-cases) utilize a CLDNN architecture, shown in Figure 4.11, which represents a more complex architecture than the small and simple CNN used in the synthetic dataset experiments. The CLDNN architecture uses both convolutional and LSTM layers, in addition to batch normalization, and could more reasonably be used in real-world systems. In [4], CLDNNs outperformed baseline CNNs, LSTMs, Inception modules, and residual networks for an AMC task, and have since been used to perform AMC throughout the literature [8, 10, 12, 116]. In this
dissertation, the use of this architecture is extended to the SEI use-case as well. Again, 128 raw IQ samples (approximately 16 symbols worth of samples) are used as input to the CLDNN architecture, fed to the network in a (1, 2, 128) tensor. The CLDNN architecture begins with three convolutional layers, each with 50 kernels of size (1, 7) and zero padding to maintain the input size. Each convolutional layer is followed by a ReLU activation function and Batch Normalization layer. The outputs of the first and third convolutional block, post Batch Normalization, are then concatenated along the channel dimension forming a (100, 2, 128) tensor, reshaped, preserving the time dimension, to form a (200, 128) tensor, and passed through a single LSTM layer with $n$ hidden cells, where $n$ is the number of output classes. The output of the LSTM layer is then flattened, passed through a fully-connected linear layer with 256 nodes, a ReLU activation function, and a Batch Normalization layer. Finally, the output layer is fully-connected with $n$ nodes with a Softmax activation function.

4.3 Model Training & Evaluation

For all model architectures, use-cases, and dataset types, the model pre-training and TL process remains the same and is shown in Fig. 4.12. The pre-training and TL hyperparameters (i.e. learning rate, weight decay, total number of epochs) remain the same for all AMC experiments, as well as all SEI experiments, but differ between the two, as discussed below.

4.3.1 Pre-Training

For pre-training, all training datasets contained 5000 examples per class, and all validation datasets contained 500 examples per class. These dataset sizes, called the “full”-sized data-subsets from here on out, are consistent with [18] and adequate to achieve consistent
4.3. MODEL TRAINING & EVALUATION

Figure 4.12: A system overview of the model pre-training, TL, and model evaluation processes used in this work.

convergence. All models were trained using the Adam optimizer \[66\] and Cross Entropy Loss \[117\], for a total of 100 epochs. The AMC models were trained with the PyTorch default hyper-parameters \[118\] (a learning rate of 0.001, without weight decay), while the SEI models were trained with a learning rate of 0.0001, also without weight decay. To prevent overfitting, a checkpoint was saved after the epoch with the lowest validation loss, and was reloaded at the conclusion of the 100 epochs.

### 4.3.2 Transfer Learning

This work examines both head re-training and model fine-tuning methods. For both head re-training and model fine-tuning, “limited”-sized data-subsets are used that contain 500 examples per class in the training dataset, and 50 examples per class in the validation dataset, representing a smaller sample of available target data (10% of the source dataset size). Both TL methods used the Adam optimizer and Cross Entropy Loss, with checkpoints saved at the lowest validation loss. All AMC models were trained with learning rate of 0.0001
and all SEI models were trained with a learning rate of 0.00001, each an order of magnitude smaller than used during pre-training. During head re-training, only the final layer of the model was trained, while the rest of the model’s parameters were frozen. During fine-tuning, the entire model was trained.

### 4.3.3 Baseline Models

Throughout this dissertation, the performance of the TL models is compared to baseline models trained on both the “full”-size and “limited”-size data-subsets. These baseline models were from random initialization using the same training hyper-parameters described for pre-training (the Adam optimizer, Cross Entropy Loss, etc.).

### 4.3.4 Evaluation

**Top-1 Accuracy**

For all experiments performed herein, top-1 accuracy is used as the performance metric throughout this dissertation, and is used to compare the performance of different source models transferred to a single target dataset, as well as to the baseline models described above. Top-1 accuracy is calculated as

\[
\text{accuracy} = \frac{\# \text{ of correct classifications}}{\text{total } \# \text{ of classifications}} \quad (4.3)
\]

**Multinomial-based Decision Aggregation**

For the experiments performed on the captured dataset, a multinomial-based decision aggregation method is used to integrate decisions on successive examples decisions into a single,
generally higher, confidence result using 10 successive examples per AMC decision and 100 successive examples per SEI decision. This aggregation helps mitigate decreases in performance caused by the added complexity of the captured data, and makes the performance trends more clear across the parameters-of-interest. The SEI use-case required more successive examples per aggregated decision because the distinguishing features for SEI are smaller variations in the raw IQ, compared to AMC, that become more apparent over longer durations. The addition of the multinomial-based decision aggregation scheme does not presume unreasonable amounts of data per decision: approximately 160-320 symbols worth of data for the AMC use-case and approximately 1600 symbols worth of data for the SEI use-case. Assuming a 250kHz sample rate and 8 samples per symbol, this corresponds to only 0.005-0.01 seconds worth of capture per AMC decision or 0.05 seconds worth of capture per SEI decision.

The multinomial-based classification decision framework is derived as follows:

Consider a pre-trained NN with $K$ possible output classes. The assumed output of the network for each input frame is a classification decision representing a choice of one possible output state from a discrete selection of $K$ possible classes. This classification decision can be re-interpreted for a single input frame as a $K$-tuple of binary outputs $\{c_1, c_2, ..., c_K\}$, where $c_j \in \{0,1\}$ and $\sum c_j = 1$. When a total of $N$ input frames are processed by the NN, the stream of $N$ outputs are accumulated over time, such that $c_j$ terms represent a short-term count of classification occurrences. The probability of selecting any specific combination $\{n_1, ..., n_K\}$, without replacement, of $N$ outputs from the NN is shown in (4.4), where $\{p_{m1}, p_{m2}, ..., p_{mK}\}$ represents the conditional probabilities of the NN producing a chosen output classification decision $\{1, ..., K\}$, given knowledge of the actual classification state $m$. These conditional probabilities $\{p_{m1}, p_{m2}, ..., p_{mK}\}$ may also be viewed as a row of entries for the classifier’s confusion matrix.
\[
P(c_1 = n_1, \ldots, c_K = n_K | m) = \frac{N!}{n_1! \cdots n_k!} p_{m1}^{n_1} \cdots p_{mk}^{n_k}
\]

The goal of determining an aggregate classification decision on \(N\) input frames then translates to the subset of all possible multinomial selections since the decision metric applicable for a classification decision scenario is that of the classification chosen (class \(j\), represented by \(c_j\)) most often; i.e.,

\[
\max_j \{c_j\} = \begin{cases}
  j = m : \text{Accurate decision} \\
  j \neq m : \text{Error}
\end{cases}
\]

Stated differently, an accurate decision occurs when \(c_m > \max_j c_j, j \neq m\). Additionally, in the scenario where \(c_m = \max_j c_j, j \neq m\), a random guess based on the leading states leads to a correct classification with a probability of \(\frac{1}{K}\). Then, the aggregate probability for selecting the correct overall classification decision from a set of \(N\) possible outputs states is the summation of multinomial state probabilities:

\[
P(c_m \geq \max_{j \neq m} c_j) > \sum_{n_m > n_l \forall l \neq m} \left( \frac{N}{\bar{n}} \right)^K \prod_{l=1}^{K} p_{nl}^{n_l} + \frac{1}{K} \sum_{n_m = \max_{n_l \neq m}} \left( \frac{N}{\bar{n}} \right)^K \prod_{l=1}^{K} p_{nl}^{n_l}
\]

To provide a general example that demonstrates the technique in more detail, consider a feed-forward NN with 4 possible outputs (\(K = 4\)) and a test accuracy of 34\% (i.e. \(p_{11} = 0.34\)), and the number inputs varying between 0 and 100 (i.e. \(N \in \{1, 2, \ldots, 100\}\)). Normally, a test accuracy of 34\% is discounted as near useless, because, for each individual input frame, the correct classification accuracy is only marginally better than blind guessing (25\%), so little confidence can be given to single output decisions. However, if we consider the following three scenarios such that the three error conditions are equiprobable (scenario 1), concentrated
4.3. Model Training & Evaluation

Figure 4.13: The aggregate classification accuracy of three example NN using the multinomial framework.

over two of the remaining three states (scenario 2), or varied over two of the remaining three states (scenario 3), the application of the proposed multinomial framework over varying values of $N$ yields the aggregate classification accuracies shown in Figure 4.13.

Scenario 1: $\{p_{11}, p_{12}, p_{13}, p_{14}\} = \{34\%, 22\%, 22\%, 22\%\}$

Scenario 2: $\{p_{11}, p_{12}, p_{13}, p_{14}\} = \{34\%, 32\%, 32\%, 2\%\}$

Scenario 3: $\{p_{11}, p_{12}, p_{13}, p_{14}\} = \{34\%, 15\%, 22\%, 29\%\}$

As expected, all three scenarios start with a classification accuracy of exactly $p_{11} = 0.34$. The oscillatory behavior for small numbers of output frames ($N < 10$) is due to the overall realization that error cases are in fact more likely than the correct classification, and thus the small population of outputs skewed probability that includes a higher density of ties ($c_m = \max_{j \neq m} c_j$) that are approximated as random guessing (25%). The difference between each of the three scenarios results from relative magnitudes of the correct classification and the highest density of error state probability. The extremely slow increase of Scenario 2 is
representative of the near random guessing between three of the four states, while the quick convergence of Scenario 1 is indicative of the maximum separation between an assumed equiprobable error condition:

\[
\frac{p_{mm}}{\max_{j \neq m} p_{mj}} = \frac{p_{mm}}{\frac{1}{N-1}(1 - p_{mm})} = \frac{(N - 1)p_{mm}}{(1 - p_{mm})}
\]

The middle state, Scenario 3, has more evenly balanced error probabilities, indicative of more general training conditions for most classifiers, and converges quickly to higher classification accuracies, but not as quickly as the maximum possible rate based on Scenario 1. The rate of convergence for these three scenarios is fundamentally a ratio of large summations without common terms, making closed form reductions impractical, even when applying assumptions as described with Scenario 1. It should be noted that a final scenario, not shown in Fig. 3, occurs when the error density of any other state exceeds that of the correct classification - in such a scenario, the aggregate classification accuracy actually decays from \(p_{11}\), making the proposed aggregation process counter-productive.

Two key simplifying assumptions are made in the proposed framework, both of which are fulfilled in this research setting: statistical independence of the input and the presence of a single observable. First, the sampling process should not induce a temporal correlation to the decisions that is different from the actual signal-of-interest, such as might occur with overlapping time windows as single frames. In all experiments performed herein, buffers are placed between examples/frames, as shown in Figure 4.8, fulfilling this assumption. Second, one consistent observable should be present for classification throughout the duration of the \(N\) input frames. This assumption eliminates the case where classification decisions overlap in time, and is most accurate for slowly changing environments and/or signals. This also assumes any non-linear effects that transform the confusion matrix entries (i.e. a fading
channel) are also slowly changing such that they occur on a timeline greater than $N$ input frames. Finally, an assumption is made that the chosen classes of the CNN-based classifier are disjoint (i.e., all probabilities add to 1), eliminating ordinal or hierarchical classification structures. These assumptions are reasonably accurate in the presence of randomly occurring background noise, without the presence of periodic interferers, as is the case for the captured dataset used herein.

4.4 Real-World Considerations

Before investigating TL performance in the next chapters, the following subsections discuss a few real-world considerations to set the stage for the remainder of the dissertation. Given the emphasis on real-world effects, all results presented in these subsections use the captured dataset. First, subsection 4.4.1 discusses how the Tx/Rx hardware, CF, and channel environment impact baseline performance, independent of the use of TL. These results begin a discussion of what factors impact the difficulty of an RFML problem, which in turn impacts performance. Next, subsection 4.4.2 discusses how CFO impacts both baseline and TL performance, setting the precedent for using CFO correction in the remainder of the experiments. Subsection 4.4.3 presents average training times for each model trained in this work, including the full/limited-data baseline models, pre-trained models, and transferred models. Finally, subsection 4.4.4 shows that models trained for one domain generally do not directly transfer to a different domain, further motivating the use of TL.
4.4.1 Impacts of Tx/Rx Hardware, CF, and Channel Environment on Baseline Performance

The full-data and limited-data baseline top-1 accuracies for the AMC models trained on the captured data transmitted by different non-overlapping groups of Txs, all received on collection node 1 at 346.3MHz at the partial occlusion location, is shown in Table 4.2. Using the full size dataset, there is little discrepancy between the performance across these three groups of Txs, with Tx Group 2 slightly underperforming in comparison to Tx Groups 1 and 3. However, using the limited size dataset, there is a much wider range of performances. In other words, given enough data, the individual Txs used herein have little effect on AMC performance.

It should be noted that in this experiment the make and model of each Tx is the same in all three groups. Unless otherwise specified, the following models are trained on the data captured at 346.3MHz using collection node 1 and all Txs at the partial occlusion location. Txs of different make, model, and quality is likely to have a bigger impact on AMC performance.

The full-data and limited-data baseline top-1 accuracies for both the AMC and SEI models trained on the data captured by each collection node is shown in Table 4.3. Compared to the baseline accuracies across Tx group, there is more variation across changes in collection node, especially when the limited size datasets are used during training and for the SEI use-case.
### 4.4. Real-World Considerations

<table>
<thead>
<tr>
<th>Collection Node</th>
<th>AMC Full Baseline</th>
<th>AMC Limited Baseline</th>
<th>SEI Full Baseline</th>
<th>SEI Limited Baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN 1</td>
<td>0.992</td>
<td>0.740</td>
<td>0.999</td>
<td>0.568</td>
</tr>
<tr>
<td>CN 2</td>
<td>0.992</td>
<td>0.919</td>
<td>0.753</td>
<td>0.295</td>
</tr>
<tr>
<td>CN 3</td>
<td>0.918</td>
<td>0.545</td>
<td>0.869</td>
<td>0.320</td>
</tr>
</tbody>
</table>

Table 4.3: The full-data and limited-data baseline top-1 accuracies for the AMC and SEI models trained on data captured by different collection nodes.

<table>
<thead>
<tr>
<th>CF</th>
<th>AMC Full Baseline</th>
<th>AMC Limited Baseline</th>
<th>SEI Full Baseline</th>
<th>SEI Limited Baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td>346.3MHz</td>
<td>0.992</td>
<td>0.740</td>
<td>0.999</td>
<td>0.568</td>
</tr>
<tr>
<td>416.4MHz</td>
<td>0.512</td>
<td>0.164</td>
<td>0.324</td>
<td>0.082</td>
</tr>
<tr>
<td>783.7MHz</td>
<td>0.829</td>
<td>0.310</td>
<td>0.391</td>
<td>0.122</td>
</tr>
</tbody>
</table>

Table 4.4: The full-data and limited-data baseline top-1 accuracies for the AMC and SEI models trained on data captured at different CFs.

Also of note, the lowest performing collection node for the AMC use-case (collection node 3) is not the same for the SEI use-case (collection node 2), highlighting that the impacts of Rx hardware on performance is task dependent.

The full-data and limited-data baseline top-1 accuracies for both the AMC and SEI models trained on data captured at different CFs is shown in Table 4.4, and the full-data and limited-data baseline top-1 accuracies for both the AMC and SEI models trained on data captured in different locations is shown in Table 4.5. Here, performance varies significantly across the different CFs and Rx locations for both the AMC and SEI use-cases, regardless of dataset size. However, the highest/lowest performing CF for the AMC use-case is consistent for the SEI use-case (346.3MHz and 416.4MHz, respectively for the different CFs, and partial occlusion and line-of-sight, respectively for the different Rx locations). Therefore, CF and Rx location are expected to be a bigger challenge to overcome in a TL setting than changes in Tx/Rx hardware, but are expected to affect both the AMC and SEI use-cases
CHAPTER 4. AN RF TL EXPERIMENTAL FRAMEWORK

<table>
<thead>
<tr>
<th></th>
<th>AMC</th>
<th></th>
<th>SEI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Full Baseline</td>
<td>Limited Baseline</td>
<td>Full Baseline</td>
</tr>
<tr>
<td>Line-of-sight</td>
<td>0.214</td>
<td>0.411</td>
<td>0.038</td>
</tr>
<tr>
<td>Partial occlusion</td>
<td>0.992</td>
<td>0.740</td>
<td>0.999</td>
</tr>
<tr>
<td>Through wall</td>
<td>0.267</td>
<td>0.042</td>
<td>0.122</td>
</tr>
</tbody>
</table>

Table 4.5: The full-data and limited-data baseline top-1 accuracies for the AMC and SEI models trained on data collected at different locations.

![AMC](image1.png) ![SEI](image2.png)

Figure 4.14: The difference between post-transfer top-1 accuracies when trained with and without CFO correction for AMC (a) and SEI (b) models. When the value is positive (to the right of the solid black line), the model trained with CFO correction outperforms the model trained without CFO correction.

Similarly. Also notable, one would typically expect the line-of-sight channel to provide the highest performance, compared to a partially occluded or through wall channel. However, as previously mentioned, the distance between the Txs and Rxs during line-of-sight capture is much farther than the distance between Txs and Rxs during the other two captures, leading to lower SNRs.
Table 4.6: Approximate training time for each model trained in this dissertation.

<table>
<thead>
<tr>
<th></th>
<th>Pre-Training &amp; Full-Baselines</th>
<th>Limited-Baselines</th>
<th>Head Re-Training</th>
<th>Model Fine-Tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Synthetic Experiments</strong></td>
<td>AMC 9 hours 25 minutes</td>
<td>25 minutes</td>
<td>20 minutes</td>
<td>30 minutes</td>
</tr>
<tr>
<td><strong>Captured Experiments</strong></td>
<td>AMC 50 minutes</td>
<td>5 minutes</td>
<td>5 minutes</td>
<td>5 minutes</td>
</tr>
<tr>
<td></td>
<td>SEI 4 hours 15 minutes</td>
<td>25 minutes</td>
<td>25 minutes</td>
<td>25 minutes</td>
</tr>
</tbody>
</table>

4.4.2 To Correct or Not to Correct CFO

Figure 4.14 shows the relative impact of CFO correction on TL performance. For the AMC use-case, CFO correction has no significant impact on performance, with most models trained with CFO correction performing within a few percentage points of those trained without. However, CFO correction does have a consistent and significant negative impact on SEI performance. Because CFO can be used as a distinguishing feature for SEI [44], performance is higher when CFO is not corrected for.

Regardless of the impact of CFO on performance, it is also important to consider what can reasonably be assumed in a real-world setting. RFML techniques are very often used in blind-Rx settings, where the Rx is not the intended recipient of the transmission and therefore does not know when and where transmissions will occur [15]. Therefore, signals-of-interest must first be detected and isolated, using RFML or traditional signal processing techniques, before any further post-processing or classification can occur. Isolation of a signal in time and frequency will almost certainly include some form of CF estimation [119] that is equivalent to the CFO correction performed herein. For this reason, CFO correction will be applied before training for all experiments in the subsequent chapters.
CHAPTER 4. AN RF TL EXPERIMENTAL FRAMEWORK

Figure 4.15: The source baseline model accuracy versus the target direct transfer accuracy for AMC and SEI models trained on the captured dataset. When the value is below the diagonal black line, the source model accuracy is greater than the target direct transfer accuracy indicating the model accuracy has degraded moving from the source domain to the target domain.

4.4.3 Training Time

Table 4.6 shows the approximate training time of each model trained in this work. All synthetic dataset experiments were trained on nodes equipped with 22-core Intel Xeon CPUs, 32Gb of RAM, and a NVIDIA Titan Xp GPU, and all captured dataset experiments were trained on nodes equipped with 20-core Intel Xeon CPUs, 80Gb of RAM, and a Tesla V100 GPU. Regardless of the hardware used during training, TL requires only 5-10% of the training time of pre-training or training from random initialization using the full-sized datasets. In other words, significant training time can be saved using TL approaches, as a result of using less data during the TL stage.

4.4.4 Testing Direct Transfer

Finally, Figure 4.15 shows the how trained models perform on in-distribution test data (x-axis, source baseline accuracy) versus how they perform on out-of-distribution test data when
4.4. Real-World Considerations

Figure 4.16: The target direct transfer accuracy versus the target post-transfer accuracy, performed using fine-tuning, for AMC and SEI models trained on the captured dataset.

TL is not performed (y-axis, target direct transfer accuracy). If the source model accuracy is greater than the target direct transfer accuracy indicating the model accuracy has degraded moving from the source domain to the target domain, the point falls below the diagonal black line. The results in Figure 4.15 show that the vast majority of models degrade when tested on out-of-distribution data, especially for the SEI use-case. Across both use-cases, performance decreases on the target test dataset versus the source test dataset for 85.68% of the models. For the AMC use-case, performance decreases for 73.50% of the models, and for the SEI use-case, performance decreases for 97.86% of the models. Additionally, on average, performance decreases by 21.85% for all models: 14.75% for the AMC models and 28.95% for the SEI models.

So, how well can TL mitigate these performance degradations? Figure 4.16 compares the target direct transfer accuracy to the target post-transfer accuracy, and shows that TL improves performance in the target domain over directly transfer in almost all cases. This then raises the question: how does TL perform in comparison to training a new model from random initialization? Figure 4.17 plots the limited-data and full-data baseline accuracies.
Figure 4.17: The target limited-data (a) and full-data (b) baseline model accuracy versus the target post-transfer accuracy, performed using fine-tuning, for AMC and SEI models trained on the captured dataset.

against the target post-transfer accuracy, and shows that when data is limited, generally TL outperforms training from random initialization, but when a sufficient data is available, TL provides little-to-no benefit. The remainder of this dissertation focuses on understanding, both qualitatively and quantitatively, when TL outperforms training from random initialization using limited amounts of data and by how much.

4.5 Discussion

This chapter has presented the experimental framework used throughout this dissertation including the synthetic and captured datasets, CNN and CLDNN model architectures, and model training and evaluation pipelines. Additionally, this chapter has examined several real-world considerations that more tangibly motivate the RF TL analysis performed in subsequent chapters. More specifically, Section 4.4 has showed how different Tx/Rx hardware, CFs, and the channels impact baseline performance to varying degrees, and that directly
transferring models trained from one domain to another can result in catastrophic performance degradation. Additionally, Section 4.4 discussed that while CFO correction does impact RF TL performance for an SEI use-case, the use of CFO correction should be assumed in RFML and RF TL settings where the Rx is typically blind. The next chapter begins the analysis of RF TL behavior, beginning with domain adaptation.
Chapter 5

An Analysis of RF Domain Adaptation Behavior

In the context of RFML, domain adaptation aims to overcome changes in data distribution caused by channel and/or Tx/Rx pairs. The task, being the use-case and label space (i.e. modulation classes, known emitters), remains the same. In the RF-specific TL taxonomy presented in Chapter 3, Section 3.3, domain adaptation is further specified by the type of domain change: environment adaptation, platform adaptation, and environment-platform co-adaptation. In this chapter, RF domain adaptation performance is examined in each of these three settings using the experiments described in Section 5.1. Performance, quantified using post-transfer top-1 accuracy, is evaluated as a function of SNR, FO, CF, and real-world channels over synthetic and captured data-types, AMC and SEI use-cases, and CNN and CLDNN architectures, each described previously in Chapter 4. The results, presented in Section 5.2, highlight real-world considerations such as the relative difficulty of performing platform adaptation versus environment adaptation and whether to use head re-training or fine-tuning. The chapter concludes with a discussion in Section 5.3 summarizing the key, actionable takeaways from the RF domain adaptation experiments, as well as remaining open questions and how they might be addressed in future work.

This chapter is adapted from the following publication:

5.1 Experiments

5.1.1 Synthetic Dataset Experiments

The examination of RF domain adaptation performance begins with experiments using synthetically generated datasets and an AMC use-case. The synthetic dataset used in this work and described in Chapter 4 was created to contain an even distribution of reasonable SNRs and FOs. Then, subsets of the data were selected according to various parameters-of-interest, artificially creating different domains. Source models were pre-trained on each domain, then transferred to the remaining domains in the experiment using both head re-training and model fine-tuning. Using this approach, performance is examined as a function of SNR alone, FO alone, and SNR and FO jointly, addressing each of the domain adaptation settings in Chapter 3, Section 3.3 – environment adaptation, platform adaptation, and environment platform co-adaptation. Sweeps over these three parameters-of-interest resulted in the training and evaluation of 4524 models over 81 data subsets. Given the training times discussed previously in Chapter 4, this corresponds to approximately 2588 hours and 50 minutes of training time.

The sweep over SNR represents an environment adaptation problem, characterized by a change in the RF channel environment such as an increase/decrease in the additive interference, \( \nu[t] \), of the channel and/or transmitting devices such as an increase/decrease in the magnitude, \( \alpha[t] \), of the transmitted signal. For this experiment, 26 source data-subsets were constructed from the synthetic master dataset containing examples with SNRs selected uniformly at random from a 5dB range sweeping from -10dB to 20dB in 1dB steps (i.e.
CHAPTER 5. AN ANALYSIS OF RF DOMAIN ADAPTATION BEHAVIOR

(a) Sweep over SNR.

(b) Sweep over FO.

(c) Sweep over SNR and FO.

Figure 5.1: The parameter-of-interest range for each synthetic domain adaptation data subset.

[-10dB, -5dB], [-9dB, -4dB], ..., [15dB, 20dB]), as shown in Figure 5.1a. FO was selected uniformly at random between [-5%, 5%] of sample rate. This SNR sweep yielded 26 baseline models and 26 pre-trained source models, each of which was transferred to the remaining 25 target data-subsets, for a total 650 models transferred using head re-training and 650 models transferred using fine-tuning.

The sweep over FO represents a platform adaptation problem, characterized by a change in the Tx and/or Rx devices such as an increase/decrease in $\omega_\Delta[t]$ due to hardware imperfections or a lack of synchronization. For this experiment, 31 source data-subsets were constructed from the synthetic master dataset containing examples with FOs selected uniformly at random from a 5% range sweeping from -10% of sample rate to 10% of sample rate.
in 0.5% steps (i.e. [-10%, -5%], [-9.5%, -4.5%], ..., [5%, 10%]), as shown in Figure 5.1b. SNR was selected uniformly at random between [0dB, 20dB]. This FO sweep yielded 31 baseline models and 31 pre-trained source models, each of which was transferred to the remaining 30 target data-subsets, for a total of 930 models transferred using head re-training and 930 models transferred using fine-tuning.

The sweep over both SNR and FO represents an environment platform co-adaptation problem, characterized by a change in both the RF channel environment and Tx/Rx devices. For this experiment, 25 source data-subsets were constructed from the synthetic master dataset containing examples with SNRs selected uniformly at random from a 10dB range sweeping from -10dB to 20dB in 5dB steps (i.e. [-10dB, 0dB], [-5dB, 5dB], ..., [10dB, 20dB]) and with FOs selected uniformly at random from a 10% range sweeping from -10% of sample rate to 10% of sample rate in 2.5% steps (i.e. [-10%, 0%], [-7.5%, 2.5%], ..., [0%, 10%]), as shown in Figure 5.1c. This SNR and FO sweep yielded 25 baseline models and 25 pre-trained source models, each of which was transferred to the remaining 24 target data-subsets, for a total of 600 models transferred using head re-training and 600 models transferred using fine-tuning.

5.1.2 Captured Dataset Experiments

The results of these initial experiments on synthetic data are then verified and extended using captured data, an additional NN architecture, and both AMC and SEI use-cases:

To examine how the Rx, propagation environment, and CF independently and jointly impact RF TL behavior, 27 data-subsets are constructed, each captured using different Rxs, Rx locations, and/or CFs, holding only the Tx hardware constant. The impacts of propagation environment alone on RF TL performance are identified by examining performance across changes in Rx location (creating different channels), while holding Tx and Rx hardware
and CF constant. As previously discussed in Chapter 4 Section 4.1.3, three different Rx locations are present in the master captured dataset: an in-room line-of-sight location, an in-room partial occlusion location, and an adjacent room location. The impacts of CF alone on RF TL performance are identified by examining performance across changes in CFs, while holding the Tx and Rx hardware and Rx location constant. Three different CFs are present in the master captured dataset: 346.3MHz, 416.4MHz, and 783.4MHz. The impacts of the Rx hardware alone on RF TL performance are identified by examining performance across changes in Rx hardware, while holding the Tx hardware, CF, and Rx location constant. Three different collection nodes are used in the construction of the master captured dataset; two host USRP B200s, and the third hosts a USRP E310. The impacts of propagation environment, CF, and Rx hardware, together, on RF TL performance are identified by examining performance across changes in all three of these parameters. These experiments are performed for both the AMC and SEI use-cases, resulting in the training of 54 baseline models, 54 pre-trained source models, 1404 head-retrained models, and 1404 fine-tuned models for each use-case.

Additionally, the impact of the Tx hardware on RF TL performance is isolated by creating 4 data-subsets from the captured master dataset, each transmitted by either Tx Group A, Tx Group B, Tx Group C, or All Tx, where Tx Groups A/B/C each contain 10 Txs randomly selected without replacement. For this experiment, all Txs are co-located, and the Rx hardware, Rx location, and CF are constant between data-subsets, using only collection node 1 (hosting a USRP B200) at the in-room partial occlusion location capturing at 346.3MHz. This experiment is performed for the AMC use-case only, as changing the Tx hardware in an SEI setting would change the task. Therefore, experiments examining the impact of Tx hardware on RF TL performance in an SEI setting are discussed in the next chapter on Sequential Learning. Having already trained baseline and source models for the All Tx setting
5.2. Results

during the previous experiment, this experiment resulted in the training of three additional baseline models, 3 additional pre-trained source models, 12 head re-trained models, and 12 fine-tuned models.

Given the training times discussed previously in Chapter 4, the captured domain adaptation experiments required an additional 857 hours of training time.

5.2 Results

5.2.1 Head Re-Training vs. Fine-Tuning

Before examining TL performance in comparison to the baselines in the following subsections, Figures 5.2 and 5.3 plot the difference between post-transfer top-1 accuracy achieved using head re-training and fine-tuning such that positive values correspond to better fine-tuning performance and negative values correspond to better head re-training performance for the synthetic and captured experiments, respectively. Across both the synthetic and captured dataset experiments, results show that head re-training is as effective, if not more effective, than fine-tuning when TL outperforms the limited-data baseline. In other words, the trends in Figures 5.2-5.3 are very similar to those seen in Figures 5.7 - 5.9 and 5.12, discussed further below. Given that head re-training is more time efficient and less computationally expensive than fine-tuning, there is a strong case for using head re-training over fine-tuning when performing RF domain adaptation.

The only conflicting result is the synthetic FO sweep experiment highlighted previously, in which fine-tuning provided better performance than head re-training when the source/target FO ranges did not overlap whatsoever. In this setting, fine-tuning likely outperforms head re-training because the features found in the early layers of the source model need modification.
Figure 5.2: The difference between post-transfer top-1 accuracies achieved using head retraining versus fine-tuning for the sweep over (a) SNR, (b) FO, and (c) SNR + FO, shown on a scale of [-0.25, 0.25] for (a) and (c) and [-0.15, 0.15] for (b). When the value is positive, fine-tuning outperforms head re-training. When the value is negative, head retraining outperforms fine-tuning.
5.2. Results

5.2. Results

(a) AMC
(b) SEI

Figure 5.3: The difference between post-transfer top-1 accuracies achieved using head re-training versus fine-tuning across all domains in the captured dataset for the AMC (a) and SEI use-cases. When the value is positive, fine-tuning outperforms head re-training. When the value is negative, head re-training outperforms fine-tuning.

to effectively identify features in the target domain. However, this trend in the simulated platform adaptation experiment did not translate to the captured data experiments. Figure 5.3 shows that for the captured data experiments, head re-training is as effective as fine-tuning for platform adaptation when TL outperforms the limited-data baseline.

5.2.2 Synthetic Dataset Experiments

The heatmaps in Figures 5.4-5.6 show the post-transfer top-1 accuracy achieved with each of the synthetic source/target dataset pairs for the sweeps across SNR alone, FO alone, and SNR and FO jointly. Results indicate that highest post-transfer performance is achieved along the diagonal of the heatmap, where the source and target domains are most similar, as all other metadata parameters are held constant. These trends are expected, as models trained on similar domains likely learn similar features, and is consistent with the general theory of TL \([79]\), as well as existing works in modalities outside of RF \([120]\).
Figure 5.4: The post-transfer top-1 accuracy for each source/target dataset pair constructed for the synthetic dataset sweeps over SNR using head re-training (a) and fine-tuning (b) to perform domain adaptation, shown on a scale of [0.0, 1.0].

Figure 5.5: The post-transfer top-1 accuracy for each source/target dataset pair constructed for the synthetic dataset sweeps over FO using head re-training (a) and fine-tuning (b) to perform domain adaptation, shown on a scale of [0.5, 0.9].
5.2. Results

(a) Head Re-Training

(b) Model Fine-Tuning

Figure 5.6: The post-transfer top-1 accuracy for each source/target dataset pair constructed for the synthetic dataset sweeps over SNR + FO using head re-training (a) and fine-tuning (c) to perform domain adaptation, shown on a scale of [0.0, 1.0].
CHAPTER 5. AN ANALYSIS OF RF DOMAIN ADAPTATION BEHAVIOR

(a) Head Re-Training  
(b) Model Fine-Tuning

Figure 5.7: The difference between post-transfer top-1 accuracy and target limited-data baseline accuracy for the sweep over SNR using head re-training (a) and fine-tuning (b), shown on a scale of [-0.35, 0.35].

Figures 5.4-5.6 also show that transfer across changes in FO is approximately symmetric, while transfer across changes in SNR are not. This behavior can be attributed to changes in the relative “difficulty” between the source and target domains. More specifically, changing the source/target SNR inherently changes the difficulty of the problem, as performing AMC in lower SNR channel environments is more challenging than performing AMC in high SNR channel environments. Therefore, the post-transfer accuracies achieved in the lower SNR target domains are lower overall than the post-transfer accuracies achieved in the higher SNR target domains. In contrast, changing the source/target FO does not make performing AMC any more or less difficult, but may require modifications to the learned features to accommodate. This can be likened to performing FO calibration, as is standard practice in Rx operations. Consequently, small changes in FO, $\omega_\Delta[t]$, in either the positive and negative direction, are expected to perform similarly. Figure 5.5 indeed shows that TL performance is approximately symmetric, with best performance closest to the diagonal where the source and target FO ranges are most closely aligned.
5.2. RESULTS

(a) Head Re-Training

(b) Model Fine-Tuning

Figure 5.8: The difference between post-transfer top-1 accuracy and target limited-data baseline accuracy for the sweep over FO using head re-training (a) and fine-tuning (b), shown on a scale of [-0.25, 0.25].

However, it is important to note that a comparatively lower post-transfer accuracy does not necessarily correspond to failed transfer, as post-transfer accuracy is also impacted by the difficulty of the target domain. In other words, transferring from an “easier” domain to a “harder” domain may result in a net loss of performance in terms of post-transfer top-1 accuracy, but TL may still outperform the limited-data baseline. Figures 5.7-5.9 present the difference between post-transfer top-1 accuracy and target limited-data baseline accuracy for the SNR and FO sweeps, such that when the difference value is positive the TL model outperforms the baseline model and vice versa. These results show that for the sweep over SNR, the TL model only outperforms the baseline near the diagonal where the source and target are very similar, including when transferring between the lowest SNR ranges. However, for the sweep over FO, the TL model outperforms the baseline for a greater number of source/target pairs.

Practically, these trends suggest that the effectiveness of RF domain adaptation increases
Figure 5.9: The difference between post-transfer top-1 accuracy and target limited-data baseline accuracy for the sweep over SNR + FO using head re-training (a) and fine-tuning (b), shown on a scale of [-0.35, 0.35].
as the source and target domains become more and more similar, and, when applicable, RF domain adaptation is more often successful when transferring from harder to easier domains when compared to transferring from easier to harder domains. For example, transferring from [-5dB, 0dB] to [0dB, 5dB] SNR is likely more effective than transferring from [5dB, 10dB] to [0dB, 5dB] SNR. Although the distance between source/target datasets (as measured by the relative change in SNR) in these two transfer scenarios is the same, the SNR increases from source to target in the first case and decreases from source to target in the second case. However, transferring from a FO range of [-9%, -4%] of sample rate to [-8%, -3%] of sample rate is likely more effective than transferring from a FO range of [-10%, -5%] of sample rate to [-8%, -3%] of sample rate, as the distance between source/target datasets (as measured by the relative change in FO) is greater in the second case.

Recalling that the sweep over SNR can be regarded as an environment adaptation experiment and the sweep over FO can be regarded as a platform adaptation experiment, the results of these synthetic dataset experiments suggest that changes in channel environment are more challenging to overcome using TL techniques than changes in Tx/Rx hardware, such that environment adaptation is more difficult to achieve than platform adaptation. While this trend is indirectly shown through the range of accuracies achieved in Figures 5.4-5.6, which is smaller for the FO sweep than the SNR sweep and SNR + FO sweep, and more directly shown in Figures 5.7-5.9. More specifically, if the source/target SNR ranges do not overlap to some degree, better performance is achieved through simply training from random initialization on the target data, even if there is a limited amount of labeled target data available. However, for overcoming changes in FO, TL is generally more successful than the limited-data baseline, especially when using model fine-tuning.
Figure 5.10: The post-transfer top-1 accuracy for each source/target dataset pair constructed for the captured dataset AMC and SEI experiments using head re-training and fine-tuning, shown on a scale of [0.0, 1.0]. The first value in each axis label refers to the CF, the second value refers to the collection node ID, and the third value refers to the Rx location.
5.2. Results

(a) AMC, Head Re-Training
(b) AMC, Model Fine-Tuning
(c) SEI, Head Re-Training
(d) SEI, Model Fine-Tuning

Figure 5.11: The post-transfer top-1 accuracy for each source/target dataset pair constructed for the captured dataset AMC and SEI experiments using head re-training and fine-tuning with the rows/columns sorted by limited-data baseline accuracy. The domain with the lowest baseline accuracy is on the upper/left and domain with the highest baseline accuracy on the bottom/right.
5.2.3 Captured Dataset Experiments

The heatmaps in Figure 5.10 shows the post-transfer top-1 accuracy achieved with each of the captured source/target dataset pairs for the AMC and SEI use-cases using both head re-training and model fine-tuning. While the synthetic dataset experiments provided results with clear and straightforward trends, the added complexity and real-world effects present in the captured dataset make deciphering the trends in Figure 5.10 more challenging. Notably, TL performance does not seem to be dictated by CF, Rx ID, or Rx location individually, and baseline performance varies significantly from domain to domain. That is, some domains are more “difficult” than others, similar to the synthetic sweep over SNR discussed previously.

Taking this into consideration, Figure 5.11 shows the same post-transfer top-1 accuracy values of Figure 5.10, but with the rows and columns sorted in order of the limited-data baseline accuracy, with the lowest baseline accuracy on the upper/left and the highest baseline accuracy on the bottom/right. The simple act of sorting by baseline accuracy yields more clear trends in the data very similar to that seen in the synthetic sweep over SNR (Figures 5.4 and 5.6), seemingly according to some notion of domain “difficulty.” More specifically, accuracy generally increases as one moves down and/or to the right in the plot, but not according to specific CFs, Rx IDs, or Rx locations, in particular. Unlike for the synthetic dataset experiments, however, Figure 5.12 shows that for the captured dataset AMC experiments, TL outperforms the limited-data baseline in most settings.

Given these results, the primary question is the cause of the relative differences in domain “difficulty.” While the synthetic dataset experiments isolated one parameter-of-interest and completely controlled for all other variables, the domains created in the captured dataset experiment are composed of three different parameters (CF, Rx ID, and Rx location), each of which directly or indirectly impact values such as CFO, signal/noise power, and SNR. The
5.2. Results

(a) AMC, Head Re-Training
(b) AMC, Model Fine-Tuning
(c) SEI, Head Re-Training
(d) SEI, Model Fine-Tuning

Figure 5.12: The difference between post-transfer top-1 accuracy and target limited-data baseline accuracy for each source/target dataset pair constructed for the captured dataset AMC and SEI experiments using head re-training and fine-tuning, shown on a scale of [-0.5, 0.5]. The first value in each axis label refers to the CF, the second value refers to the collection node ID, and the third value refers to the Rx location.
results in Figures 5.11 and 5.12 suggest that neither CF, Rx ID, or Rx location independently encourages or prevents transfer to other CFs, Rx IDs, or Rx locations, as there are no apparent trends associated with any one CF, Rx ID, or Rx location in particular. Similarly, Figures 5.13 and 5.14 show that changes in groups of Tx alone also does not have a significant impact on TL performance the AMC use-case. While there is some slight variation in performance across the Tx groups, post-transfer accuracy is consistently high and transfer is good between all source/target pairs, always outperforming the limited-data baseline for all source/target pairs.

TL performance is seemingly better predicted by the relative performance of the source and target baseline models, with transfer more likely to occur when the source baseline model outperforms the target baseline model. Given that CFO is corrected before training and does not appear to be statistically different between domains, the most apparent hypothesis for the cause of the relative performance differences between domains is a change in SNR, though is almost certainly not the sole contributor. Figure 5.15 shows the distribution of SNRs for all data points in the bottom three performing domains and the top three performing domains for both the AMC and SEI use-cases. On average, the data points from the higher performing domains have an approximately 20dB higher SNR than those from the lower performing domains. However, the SNR of all data points in the captured dataset are sufficiently high that a decrease of 10dB SNR should not result in such severe performance degredations.

A secondary hypothesis is that performing TL using a larger target dataset size (i.e. 20-50% of the source dataset size) might minimize the relative performance differences between domains, effectively smoothing out the post-transfer top-1 accuracy plots in Figures 5.10 and 5.11. If this were the case, it may indicate that despite using three different Rx locations, the SNRs in each channel created are not sufficiently different and/or low enough to yield
5.2. Results

(a) Head Re-Training  
(b) Model Fine-Tuning

Figure 5.13: The post-transfer top-1 accuracy across changing groups of Txs for the captured dataset AMC use-case using head re-training and fine-tuning, shown on a scale of [0.9, 1.0].

(a) Head Re-Training  
(b) Model Fine-Tuning

Figure 5.14: The difference between post-transfer top-1 accuracy and target limited-data baseline accuracy across changing groups of Txs for the captured dataset AMC use-cases using head re-training and fine-tuning.
the performance trends seen in the synthetic dataset experiments across SNR. Although this dissertation has focused on varying only the parameters within the RF data while fixing the TL methods, follow on work could address this question by (1) experimenting with RF data captured in a wider variety of conditions and (2) experimenting with larger amounts of data.

Experimentation with larger datasets would also likely improve the results of the SEI experiments. In comparison to the AMC use-case, post-transfer accuracy is lower in the SEI use-case, as a result of the more challenging task and increased number of output classes, TL outperforms the limited-data baseline less frequently and to a far lesser degree. Despite this, the performance trends remain the same between the AMC and SEI use-cases. Increasing the source and/or target dataset sizes, and potentially introducing further regularization or learning rate scheduling could improve the performance of the SEI approach and should be investigated in future work.
5.3 Discussion

This chapter has focused on the impacts of SNR, FO, CF, Rx/Tx ID, and channel on RF domain adaptation performance, as quantified by post-transfer top-1 accuracy compared to the limited data baseline models. In total, 112 domains were created using the synthetic and captured datasets, and 7470 models were trained. Given the breadth of signal types and parameters observed, and experimentation over both synthetic and captured data, AMC and SEI use-cases, and two different NN architectures, some generalized conclusions can be drawn regarding TL performance as a function of changes in the propagation environment and Tx/Rx hardware:

First, when performing RF domain adaptation head re-training provides the similar or better performance than model fine-tuning while being more time and computationally efficient, and is therefore the preferred TL method. Additionally, results indicate that though domain similarity is important in predicting the success of TL, the relative difficulty of the source and target domains is perhaps more important. When the source and target domains are of similar difficulty, TL performance is approximately symmetric. When one domain is more difficult than the other, TL performance is more successful from the harder domain to the easier domain. Most practically, this means that performing platform adaptation is generally successful and symmetric, so long as the source and target platforms are of similar quality, and when performing environment adaptation or environment platform co-adaptation, performance is at least somewhat dependent upon the SNRs of the source/target domains. However, TL performance and domain “difficulty” are not dictated by CF, Rx ID, or Rx location individually, and though seemingly correlated with SNR, there also seems to be other contributing factors. While this notion of domain “difficulty” is ill-defined, the quantification of RF dataset similarity and model transferability, discussed in Chapter 7, provides further insight.
For the synthetic dataset experiments, TL only provided performance benefits over the limited-data baseline when the source and target domains were similar. However, for the more realistic captured dataset AMC experiments, TL outperforms the limited-data baseline in most settings. Despite the underwhelming overall performance of the SEI models, these performance trends remain the same across the AMC and SEI use-cases. These results indicate that TL should be used when training data is limited in the target domain, to achieve maximal performance.

The added complexity of the real-world data has also raised several additional research questions and opened directions for future work. First, in the captured dataset experiments, domain difficulty varied significantly, and was not solely predicted by CF, Rx ID, or Rx location. While changes in SNR seemed to be a contributing factor to domain difficulty, it is likely not the only factor. Second, while the SEI experiments showed the same trends as the AMC experiments, post-transfer accuracy was lower and TL less successful overall. Both of these open questions are also touched on in Chapter 7, but should be more thoroughly addressed in future experiments varying the source/target dataset size and using data captured in a wider variety of conditions. Additionally, domain difficulty could be further examined through experiments measuring TL performance as a function of additional parameters-of-interest, and SEI performance could be improved through increased use of regularization and/or learning rate schedulers.
Chapter 6

An Analysis of RF Sequential Learning Behavior

Sequential learning describes any TL setting in which the task changes. For example, in RF settings, sequential learning can be used to add modulation schemes to AMC models, remove emitters from SEI models, or convert an AMC model into an SEI model. In this chapter, RF sequential learning performance is evaluated across broad categories of modulation types, namely linear, frequency-shifted, and analog modulation schemes, and groups of Txs, as well as in a successive model refinement scenario, where a single modulation type or Tx is added/removed from the source dataset using the experiments described in Section 6.1. While sequential learning includes the setting where both the source/target domain and task are changing, this chapter only examines changes in task, independent of source/target domain. Experiments examining sequential learning across changes in both domain and task are left for future work, and are discussed further in Chapter 8. The results, presented in Section 6.2, highlight the importance of source/target task similarity as an indicator or RF sequential learning performance, and provide guidelines for how best to transfer between groups of modulation schemes or Txs, as well as how best to add or remove output classes from a given source model. Finally, Section 6.3 concludes the chapter, summarizes the guidelines for RF sequential learning, and briefly discusses directions for future work.

This chapter is adapted from the following publication:

6.1 Experiments

6.1.1 Synthetic Dataset Experiments

As in Chapter 5, sequential learning performance is first examined using synthetic data for an AMC use-case, then using captured data for both AMC and SEI use-cases. Two experiments are performed using the synthetic dataset: The first experiment aims to investigate TL performance across broad categories of modulation types, namely linear, frequency-shifted, and analog modulation schemes. More specifically, 5 source data-subsets were constructed from the larger master dataset containing the modulation schemes shown in Figure 6.1. For each data-subset in this experiment, called “Synthetic Sequential AMC,” SNR was selected uniformly at random between [0dB, 20dB] and FO was selected uniformly at random between [-5%, 5%] of sample rate. This experiment yielded 5 pre-trained source models, each of which was transferred to the remaining 4 target data-subsets, yielding 20 models transferred using

![Figure 6.1: The modulation schemes in each data-subset in the Synthetic Sequential AMC experiment.](image-url)
head re-training and 20 models transferred using fine-tuning. Additionally, 5 baseline models were trained.

The second experiment performed using synthetic data examines TL performance when a single modulation scheme was added or removed from the source task, or a model refinement scenario. More specifically, the 12 source data-subsets were constructed from the larger master dataset containing the modulation schemes shown in Figure 6.2. Again, SNR was selected uniformly at random between [0dB, 20dB] and FO was selected uniformly at random between [-5%, 5%] of sample rate. This experiment is called “Synthetic Model Refinement AMC” herein. This experiment yielded 12 pre-trained source models, each of which was transferred to the remaining 11 target data-subsets, yielding 132 models transferred using head re-training and 132 models transferred using fine-tuning. Additionally, 12 baseline
Figure 6.3: The modulation schemes in each data-subset in the Captured Model Refinement AMC experiment.

models were trained.

### 6.1.2 Captured Dataset Experiments

Three additional sequential learning experiments are performed using the captured dataset. For the AMC use-case, only the model refinement experiment is conducted because of the limited number of modulation schemes available. The 4 source data-subsets constructed contain the modulation schemes shown in Figure 6.3. For this experiment, called “Captured Model Refinement AMC,” the Tx and Rx hardware, Rx locations, and CF, were held constant. Four baseline models, 4 pre-trained source models, 12 head-retrained models, and 12 fine-tuned models were trained.

For the SEI use-case, an additional two experiments are conducted mirroring those done with the synthetic dataset. The first experiment aims to investigate TL performance across 3 non-overlapping sets of emitters (Tx Group A, Tx Group B, Tx Group C), as well as to the full set of available Tx. For this experiment, called “Captured Sequential SEI,” 4 baseline models, 4 pre-trained source models, 12 head-retrained models, 12 fine-tuned models were trained. The second experiment examines TL performance when 1-10 Txs are added or removed from the source task, or a model refinement scenario. For this experiment, called “Captured Model Refinement SEI,” 11 baseline models, 11 pre-trained source models,
6.2 Results

6.2.1 Sequential Learning Across Signal Types

Figure 6.4 shows the post-transfer top-1 accuracy for each source/target dataset in the Synthetic Sequential AMC experiment. Results indicate that the subsets containing only a single type of modulation scheme (the analog, frequency-shifted, and linear subsets) don’t transfer well between one another, and also don’t transfer well to the subsets which contain multiple types of modulation schemes (the small and all subsets). Meanwhile, the small and all subsets transfer fairly well to the analog, frequency-shifted, and linear subsets. These results are verified by the results shown in Figure 6.5 which presents the difference between post-transfer top-1 accuracy and the limited-data baseline target models, and shows that TL only increases performance over the baseline models when there is significant overlap between the modulation schemes in each subset. The same trends are shown for both TL methods tested, head re-training and model fine-tuning, a topic discussed further in Sub-Section 6.2.4.

These results are expected considering the general setting in which TL is beneficial: when the source and target are “similar.” When no similar signal types between source/target there is little-to-no benefit to using TL, such as when attempting transfer between the analog, frequency-shifted, and linear subsets. However, because the small and all subsets contain at least one modulation scheme from each of the analog, frequency-shifted, and linear subsets, the pre-trained source model has some prior knowledge of each category of modulation schemes from which to build. Practically, these results indicate that for this AMC use-case,
Figure 6.4: The post-transfer top-1 accuracy for each source/target dataset pair constructed for the Synthetic Sequential AMC experiment using head re-training (a) and fine-tuning (b), shown on a scale of [0.35, 1.0].

Figure 6.5: The difference between post-transfer top-1 accuracy and target limited-data baseline accuracy for the Synthetic Sequential AMC experiment using head re-training (a) and fine-tuning (b), shown on a scale of [-0.45, 0.45].
6.2. Results

TL is only beneficial when similar signal types are present in the source and target datasets.

6.2.2 Sequential Learning Across Groups of Txs

Figures 6.6 and 6.7 show the post-transfer top-1 accuracy and difference between post-transfer top-1 accuracy and target limited-data baseline accuracy for each source/target dataset in the Captured Sequential SEI experiment. In contrast to the results in Figure 6.4, there is a significant performance difference between the models transferred using head re-training versus fine-tuning. While Figure 6.6a shows that the SEI models transferred using head re-training generally do not transfer well to other groups of Txs, Figure 6.6b shows much better transfer in the same setting using fine-tuning. It should be noted that the increases in performance when the source and target are the same (i.e. along the diagonal) is due to the 10x increase in training data between the baseline and pre-trained models.

Focusing on the model fine-tuning results in Figures 6.6b and 6.7b, the model trained on all Txs transfers well to the non-overlapping Tx groups A, B, and C. This trend is similar to that shown for the Synthetic Sequential AMC experiment (Figure 6.4). The model trained on all Txs has prior knowledge about each of the groups A, B, and C because Tx groups A, B, and C are each subsets of all Txs. Interestingly, all of the sub-groups transfer well to Tx groups A and B, but none of them transfer particularly well to Tx group C. This further reinforces the concept that TL behavior is not symmetrical (i.e. txC transfers to txA, but txA does not transfer to txC). This would also suggest that the Txs in groups A and B are more similar to one another than the Txs in group C, despite the fact that every Txs in this dataset is of the same make and model further motivating the ability to quantify similarity, explored in Chapter 7.
CHAPTER 6. AN ANALYSIS OF RF SEQUENTIAL LEARNING BEHAVIOR

Figure 6.6: The post-transfer top-1 accuracy for each source/target dataset pair constructed for the Captured Sequential SEI experiment using head re-training (a) and fine tuning (b), shown on a scale of [0.0, 1.0].

Figure 6.7: The difference between post-transfer top-1 accuracy and target limited-data baseline accuracy for the Captured Sequential SEI experiment using head re-training (a) and model fine-tuning (b), shown on a scale of [-0.85, 0.85].
6.2.3 Sequential Learning For Successive Model Refinement

Figures 6.8-6.13 present the post-transfer top-1 accuracy and the difference between the post-transfer top-1 accuracy and target limited-data baseline accuracy for the Synthetic Model Refinement AMC, Captured Model Refinement AMC, and Captured Model Refinement SEI experiments. Across all model refinement experiments, results indicate that it is easier to remove output classes during TL than it is to add output classes, as evidenced by higher performance in the upper triangle of the heatmap in Figures 6.8, 6.10b, and 6.13b, as well as the significant performance benefits over the target baseline models shown in Figures 6.9, 6.11b, and 6.13b. Again, the increase in performance along the diagonal of the heatmaps in Figures 6.12 and 6.13 is due to the 10x increase in training data between the baseline and pre-trained models.

This behavior is consistent with results given in [98] and intuitive, as it is easier to forget or disregard prior knowledge than to acquire new knowledge during transfer. More specifically, by pre-training on a larger subset of signal types or Tx IDs, the source model has already learned features to identify all of the modulation classes or Txs in the target task. In fact, the source model has likely learned more features than necessary to perform the target task, and could undergo feature pruning in order to reduce computational complexity. It should also be noted that the task gets easier as output classes are removed, further contributing to the trend. Practically, these results dictate that one should utilize a source task that encompasses the target task, when possible.

6.2.4 Head Re-training vs. Fine Tuning

Figure 6.14 presents the difference between post-transfer top-1 accuracies achieved using head re-training versus fine-tuning for the Synthetic Sequential AMC and Synthetic Model...
Figure 6.8: The post-transfer top-1 accuracy for each source/target dataset pair constructed for the Synthetic Model Refinement AMC experiment using head re-training (a) and fine-tuning (b), shown on a scale of [0.6, 0.9].

Figure 6.9: The difference between post-transfer top-1 accuracy and target baseline accuracy for Synthetic Model Refinement AMC using head re-training (a) and fine-tuning (b), shown on a scale of [-0.25, 0.25].
6.2. Results

(a) Head Re-Training

(b) Model Fine-Tuning

Figure 6.10: The post-transfer top-1 accuracy for each source/target dataset pair constructed for the Captured Model Refinement AMC experiment using head re-training (a) and fine tuning (b), shown on a scale of $[0.6, 1.0]$.

(a) Head Re-Training

(b) Model Fine-Tuning

Figure 6.11: The difference between post-transfer top-1 accuracy and target limited-data baseline accuracy for the Captured Model Refinement AMC experiment using head re-training (a) and model fine-tuning (b), shown on a scale of $[-0.25, 0.25]$. 
Figure 6.12: The post-transfer top-1 accuracy for each source/target dataset pair constructed for the Captured Model Refinement SEI experiment using head re-training (a) and fine tuning (b), shown on a scale of [0.0, 1.0].

Figure 6.13: The difference between post-transfer top-1 accuracy and target limited-data baseline accuracy for the Captured Model Refinement SEI experiment using head re-training (a) and model fine-tuning (b), shown on a scale of [-0.85, 0.85].
6.2. Results

(b) Synthetic Model Refinement AMC

Figure 6.14: The difference between post-transfer top-1 accuracies achieved using head re-training versus fine-tuning for (a) the Synthetic Sequential AMC and (b) Synthetic Model Refinement AMC experiments. When the value is positive, fine-tuning outperforms head re-training. When the value is negative, head re-training outperforms fine-tuning.
Figure 6.15: The difference between post-transfer top-1 accuracies achieved using head re-training versus fine-tuning for (a) the Captured Model Refinement AMC, (b) Captured Sequential SEI, and (c) Captured Model Refinement SEI experiments. When the value is positive, fine-tuning outperforms head re-training. When the value is negative, head re-training outperforms fine-tuning.
Refinement AMC experiments. For these experiments performed using synthetic data, fine-tuning outperforms head re-training only in cases where the source/target tasks are ‘less similar” such as between linear, frequency-shifted, and analog modulation types. Meanwhile, when the source/target subsets have some modulation schemes in common, head re-training performs as well, or better than, fine-tuning, but only by a small margin. However, the results for the captured dataset experiments in Figures 6.6, 6.7, 6.10, 6.11, 6.12, and 6.13 have already shown a significant difference between post-transfer accuracy using head re-training and fine-tuning. Figure 6.15 presents the difference between post-transfer top-1 accuracies achieved using head re-training versus fine-tuning for the Captured Sequential SEI and Captured Model Refinement AMC/SEI experiments, and shows that in almost all cases, fine-tuning outperforms head re-training, often by a large margin.

The synthetic data is observed in a pristine environment compared to the effects of a real-world channel. As a result, there is far more variance between examples with the “same” metadata parameters in the captured dataset compared to the synthetic dataset. This, in turn, affects not only the complexity of the task, making AMC and SEI more challenging in the real-world, but also likely affects the similarity of the source/target datasets. Results thus far have shown that in settings where the source and target domain and/or task are less similar, fine-tuning outperforms head re-training, while head re-training is typically sufficient in settings where the source and target domain and/or task are similar. Therefore, next chapter will examine metrics and methods for quantifying this notion of similarity and predicting transfer accuracy.
6.3 Discussion

This chapter has looked at RF TL performance in settings where the task changes between the source and target, but the domain stays the same. In particular, the experiments performed herein have examined sequential learning performance across broad categories of modulation types and groups of Txs, and when adding or removing one or more output class from the source task. In total, 43 tasks were created using the synthetic and captured datasets, and 594 models were trained. The training of these 594 models took just over 436 hours.

Results showed that attempting to transfer across signal types for an AMC use-case, TL is only beneficial when similar signal types are present in the source/target task. Likewise, when attempting to transfer across groups of Txs for an SEI use-case, it is beneficial for there to be some overlap between the source and target Txs groups, though transfer across disjoint groups of Txs can sometimes be achieved using fine-tuning instead of head re-training. When performing model refinement, results showed better performance when removing output classes than when adding output classes. In other words, when possible, the source task should encompass the target task. Finally, while head re-training slightly outperformed fine-tuning for the synthetic sequential experiments, model fine-tuning vastly outperformed head re-training for the captured sequential experiments. This is likely due to more significant dissimilarity between source/target caused by non-idealities of the captured data, paired with the changing tasks, resulting in the need to modify the learned features of the source model for the target task. Therefore, fine-tuning is better suited to RF sequential learning.

As previously mentioned, this chapter has only examined changes in task, independent of source/target domain. Future work includes examining sequential learning across both domain and task changes, combining the experiments of the previous two chapters. Such
experiments are outlined in detail in Chapter 8.
Chapter 7

Metrics for Predicting RF TL Performance

Throughout this dissertation, we have examined RF TL performance to better understand the impacts of the channel, Tx/Rx hardware variations, and CF impact learned behavior. Amongst the finer details discussed, the results of these experiments have highlighted that source/target domain and task similarity is key to successful TL, as is the case in other modalities. However, until this point, the term “similar” has been qualified intuitively, not quantified, and has relied upon having comprehensive metadata for the source/target domains and tasks. This chapter examines metrics and methods for source model selection and predicting RF TL behavior without knowledge of the source/target metadata. More specifically, we examine two existing transferability metrics, LEEP and LogME (described in Section 7.1.1), for selecting pre-trained source models for TL using the target dataset alone, and show how they can be used to predict TL performance (Section 7.2.2). Furthermore, a novel raw RF dataset similarity metric, based on expert-defined features and $\chi^2$ tests, is presented in Section 7.1.2 to intuitively quantify the notion of similar signal sets without the use of metadata or labels. The results, presented in Section 7.2, highlight the correlation between LEEP/LogME and RF TL performance, and show the ability to use LEEP/LogME for RF source model selection.
7.1 Definitions

TL techniques aim to further refine a pre-trained source model using a target dataset and specialized training techniques. However, not all pre-trained source models will transfer well to a given target dataset. In this dissertation, post-transfer top-1 accuracy has provided the ground truth measure of transferability by quantifying performance after TL has occurred through head re-training or fine-tuning of the source model. In the scenario where many source models are available for transfer to a target domain/task, performing TL on each source model and evaluating the post-transfer top-1 accuracy on the target dataset may be too time consuming and computationally expensive. The following subsections describe other metrics used to predict and measure TL performance without additional training.

7.1.1 Transferability Metrics

The goal of a transferability metric is to quantify how well a given pre-trained source model will transfer to a target dataset. While the area of transferability metrics is growing increasingly popular, no prior works have examined these metrics in the context of RFML. Transferability metrics developed and examined in the context of other modalities can broadly be categorized in one of two ways: those requiring partial re-training and those that do not.

Partial re-training methods such as Taskonomy [121] and Task2Vec [122] require some amount of training to occur, whether that be the initial stages of TL, full TL, or the training of an additional probe network, in order to quantify transferability. Partial re-training methods are typically used to identify relationships between source and target tasks and are useful in meta-learning settings, but are not well suited to settings where time and/or computational resources are limited. Though the computational complexity of partial re-training methods varies, it vastly exceeds the computational complexity of methods that do
not require any additional training, such as those considered herein.

This dissertation focuses on methods that do not require additional training, which typically use a single forward pass through a pre-trained model to ascertain transferability, as these methods would be most useful in a real-world RF TL setting. Methods such as these are often used to select a pre-trained model from a model library for transfer to a target dataset, a problem known as source model selection. LEEP [123] and LogME [124] were chosen for this work having outperformed similar metrics such as Negative Conditional Entropy (NCE) [125] and H-scores [126] in CV and NLP-based experiments, and for their modality agnostic design. However, several new transferability metrics developed concurrently with this work also show promise including Optimal Transport-based Conditional Entropy (OTCE) [127] and Joint Correspondences Negative Conditional Entropy (JC-NCE) [128], TransRate [129], and Gaussian Bhattacharyya Coefficient (GBC) [130], and may be examined as follow on work.

Related works examine source model ranking or selection procedures [131, 132], which either rank a set of models by transferability or select the model(s) most likely to provide successful transfer. However, source model ranking or selection methods are less flexible than transferability metrics in online or active learning scenarios. More specifically, source model ranking or selection methods are unable to identify how a new source model compares to the already ranked/selected source models without performing the ranking/selection procedure again. Related works also include methods for selecting the best data to use for pre-training [133] or during the transfer phase [80], and approaches measuring to domain, task, and/or dataset similarity [134], further examined in Section 7.1.2.
LEEP

LEEP [123] can be described as the “average log-likelihood of the expected empirical predictor, a simple classifier that makes prediction[s] based on the expected empirical conditional distribution between source and target labels,” and is calculated as

\[
T(f_S, X_T) = \frac{1}{n} \sum_{i=1}^{n} \log \left( \sum_{y_S \in Y_S} \hat{P}(x_T^i | y_S) f_S(x_T^i) y_S \right)
\]

(7.1)

such that \(f_S\) is the pre-trained source model, \(X_T\) is the target dataset, \(n\) is the number of examples in the target dataset, \(Y_T\) is the set of all target labels, \(Y_S\) is the set of all source labels. \(\hat{P}(y_T | y_S)\) is computed using \(\hat{P}(y_T, y_S)\) and \(\hat{P}(y_S)\) with

\[
\hat{P}(y_T | y_S) = \frac{\hat{P}(y_T, y_S)}{\hat{P}(y_S)}
\]

(7.2)

where

\[
\hat{P}(y_T, y_S) = \frac{1}{n} \sum_{i:y_T^i = y_T} f(x_T^i) y_S
\]

(7.3)

and

\[
\hat{P}(y_S) = \sum_{y_T \in Y_T} \hat{P}(y_T, y_S) = \frac{1}{n} \sum_{i=1}^{n} f(x_T^i) y_S
\]

(7.4)

LEEP has been shown to correlate well with transfer accuracy using image data, even when the target datasets are small or imbalanced. The metric is bounded between \((-\infty, 0]\), such that values closest to zero indicate best transferability, though the scores tend to be smaller when there are more output classes in the target task. The calculation does not make any assumptions about the similarity of the source/target input data, except that they are the same size. For example, if the source data is raw IQ data of size 2x128, then the target data must also be of size 2x128, but need not be in raw IQ format (i.e. the target data could be
in polar format. Therefore, the metric is suitable for estimating transferability when the source and target tasks (output classes) differ. However, the calculation of the metric does assume the use of a Softmax output layer, limiting the technique to supervised classifiers.

LogME

In comparison to LEEP, which measures the expected empirical distribution between the source and target labels, LogME [124] estimates the maximum evidence, or marginal likelihood, of a label given the features extracted by the pre-trained model at some layer $j$ using a computationally efficient Bayesian algorithm. Letting $y$ be the groundtruth labels of the target dataset, $X_T$, of size $n$, and $D$ be the dimensionality of the feature space $F$ extracted from the pre-trained model at layer $j$ given $X_T$ as input, LogME computed as follows: The logarithm of the evidence is computed using

$$
\text{argmax}_{\alpha, \beta} \mathcal{L}(\alpha, \beta) = \log p(y|F, \alpha, \beta)
$$

$$
= \frac{n}{2} \log \beta + \frac{D}{2} \log \alpha - \frac{n}{2} \log 2\pi - \frac{\beta}{2} \|Fm - y\|_2^2 - \frac{\alpha}{2} m^T m - \frac{1}{2} \log |A|
$$

(7.5)

where $A = \alpha I + \beta F^T F$ and $m = \beta A^{-1} F^T y$. The full derivation of Equation 7.5 can be found in [124]. Maximization of $\mathcal{L}(\alpha, \beta)$ is achieved by iteratively evaluating $m$ and

$$
\gamma = \sum_{i=1}^{D} \frac{\beta \sigma_i}{\alpha + \beta \sigma_i}
$$

(7.6)

with $\sigma$ being the singular values of $F^T F$, and updating

$$
\alpha \leftarrow \frac{\gamma}{m^T m}, \quad \beta \leftarrow \frac{n - \gamma}{\|Fm - y\|_2^2}
$$

(7.7)
7.1. Definitions

until $\alpha$ and $\beta$ converge. (Generally in 1-3 iterations.) Finally, $\arg\max_{\alpha, \beta} \mathcal{L}(\alpha, \beta)$ is scaled by $n$ to compute the average maximum log evidence of $y_i$ given $F_i$ for all $i \in \{1, \ldots n\}$, or LogME.

Like LEEP, this calculation only assumes that the source and target input data are the same size. The metric is bounded between $[-1, 1]$, such that values close to -1 indicate worst transferability and values closest to 1 indicate best transferability. LogME does not require the use of a Softmax output layer, and is therefore appropriate in un-supervised settings, regression settings, and the like. Further, LogME was shown to outperform LEEP in an image classification setting, better correlating with transfer accuracy, and has also shown positive results in an NLP setting.

7.1.2 Dataset Similarity

Consistent with the literature [79], the results of the previous chapters has shown that TL performance is generally correlated with the similarity of the source and target domains and tasks. Given that the source and target datasets characterize the respective domains and tasks, source and target domain and task similarity can be considered through the lens of dataset similarity.

In the fields of CV and NLP several dataset similarity metrics have been considered, including information theoretic measures (i.e. Kullback-Leibler Divergence (KL divergence) and Jenson-Shannon Divergence (JS divergence)) [135], higher-order measures (i.e. Maximum Mean Discrepancy (MMD)) [136], Principal Component Analysis (PCA)-based metrics [137], and the Proxy-A distance [138]. Similar metrics have demonstrated success in feature-based CR settings, but have not yet demonstrated success with raw RF data which is high-dimensional, fast-changing, and highly dependent on the underlying bit pat-
tern [3, 139, 140, 141, 142]. For example, in [139], the KL divergence between historical measurement data such as Received Signal Strength (RSS) and Signal-to-Interference-Plus-Noise Ratio (SINR) is used to estimate the similarity between femtocells. Other works have made use of the Frechet Inception Distance (FID) evaluation index and MMD between NN embeddings of spectograms [140, 141]. Several works have also examined the similarity between users in Dynamic Spectrum Access (DSA) enabled CR networks which utilize between Deep Q-Networks (DQN). Such approaches quantify similarity between each pairs of secondary users in the network using the MSE between the action-value function parameters [3]. Finally, work in [142] uses Dynamic Time Warping (DTW) to examine the similarity between frequency bands on an example-by-example basis for improving TL performance in a cross-band spectrum prediction setting. However, these existing approaches generally compare lower dimensional measurement data history or action-value function parameters with the task held constant, rely on NN embeddings, which can be expensive to compute and inconsistent between model architectures, and/or compare the similarity between individual examples rather than entire datasets.

The following metric aims to measure the similarity between two raw RF datasets through their expert-defined feature distributions, extracted from each example in the source/target datasets, and using $\chi^2$ tests. The result is a flexible and computationally efficient dataset similarity metric, bounded between 0 and 1, that is applicable to both labeled and un-labeled datasets. By focusing solely on the datasets and expert-defined features to estimate domain and task similarity, no ML/DL training/execution is required, minimizing compute. Moreover, the proposed metric is largely invariant to the size of datasets provided enough samples are available to characterize the feature distribution, as discussed in Section 7.2.3. The use of expert-defined features provides an intuitive notion of similarity, especially for traditional RF engineers who rely heavily on feature-based methods. Finally, while the metric, as out-
lined, is specific to an AMC use-case, it is easily extended to other use-cases (e.g. SEI, signal detection) and even additional modalities (e.g. CV, NLP) with appropriately chosen expert-defined features.

The proposed dataset similarity metric calculated as follows:

1. Calculate $n$ expert features for each sample in $d_0$ and $d_1$, creating feature vectors $f_{1,0}$, ..., $f_{1,n-1}$ and $f_{2,0}$, ..., $f_{2,n-1}$.

2. For each feature vector, construct a histogram $(h_{1,0}, ..., h_{1,n-1}$ and $h_{2,0}, ..., h_{2,n-1})$ such that $h_{1,0}$ is the histogram version of $f_{1,0}$, $h_{2,0}$ is the histogram version of $f_{2,0}$, and so on. Note that each histogram will have the same bin locations (start/stop).

3. Use Chi-squared tests to determine whether there is a statistically significant difference between $h_{1,0}$ and $h_{2,0}$, $h_{1,1}$ and $h_{2,1}$, ..., and $h_{1,n-1}$ and $h_{2,n-1}$, recording the $p$-values for each test ($p_0, ..., p_{n-1}$).

4. The final dataset similarity score is the weighted average of the $p$-values recorded:

$$\sum_{i=0}^{n-1} \alpha_i \cdot p_i$$

The following hyper-parameters and design options are to be chosen by the user, offering flexibility and customization:

- The number and type of expert features used in step #1. For proof of concept using an AMC use-case, the 7 time-domain features given in Table 7.1 have been chosen for their simplicity and use in existing feature-based AMC works. However, it should be noted that these features must be adapted to the problem and datasets contents. For example, transient-based features might be used in an SEI setting [143], while higher-
### Table 7.1: The instantaneous time-domain features used in this work, where \( s(t) \), \( \phi(t) \), and \( f(t) \) are the instantaneous amplitude, phase, and frequency of the example, respectively.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Formula</th>
</tr>
</thead>
</table>
| The standard deviation of the absolute value of the normalized instantaneous amplitude of the example | \[
\sigma_{\text{aa}} = \sqrt{\frac{1}{N_s} \sum_{i=1}^{N_s} a_n^2(i) - \left(\frac{1}{N_s} \sum_{i=1}^{N_s} |a_n(i)| \right)^2}
\]
| \( a_n(i) = a(i)m_a \) | \( m_a = \frac{1}{N_s} \sum_{i=1}^{N_s} a(i) \) |
| \( N_s \) is the example length |
| The standard deviation of the instantaneous phase of the example | \[
\sigma_{\text{dp}} = \sqrt{\frac{1}{N_s} \sum_{i=1}^{N_s} \phi_2(i) - \left(\frac{1}{N_s} \sum_{i=1}^{N_s} |\phi(i)| \right)^2}
\]
| The kurtosis of the instantaneous amplitude | \[
K_a = \frac{\sum_{i=1}^{N_s} (a(i) - m_a)^4}{\left(\sum_{i=1}^{N_s} (a(i) - m_a)^2\right)^2}
\]
| \( m_a = \frac{1}{N_s} \sum_{i=1}^{N_s} a(i) \) |
| \( N_s \) is the example length |
| The kurtosis of the instantaneous frequency | \[
K_f = \frac{\sum_{i=1}^{N_s} (f(i) - m_f)^4}{\left(\sum_{i=1}^{N_s} (f(i) - m_f)^2\right)^2}
\]
| \( m_f = \frac{1}{N_s} \sum_{i=1}^{N_s} f(i) \) |
| \( N_s \) is the example length |
| Spectrum symmetry | \[
P_L = \frac{\sum_{i=1}^{N_s} S(i)^2}{P_U} = \frac{\sum_{i=1}^{N_s} S(i)^2}{\sum_{i=1}^{N_s} S(i + f_c + 1)^2}
\]
| \( f_c \) is the sample number corresponding to the carrier frequency |
| \( S(t) \) is the frequency-domain representation of \( s(t) \) |
| The standard deviation of the absolute value of the normalized instantaneous frequency of the example | \[
\sigma_{\text{fn}} = \sqrt{\frac{1}{N_s} \sum_{i=1}^{N_s} f_n^2(i) - \left(\frac{1}{N_s} \sum_{i=1}^{N_s} |f_n(i)| \right)^2}
\]
| \( f_n(i) = f(i)m_a \) | \( m_a = \frac{1}{N_s} \sum_{i=1}^{N_s} f(i) \) |
| \( N_s \) is the example length |
| The standard deviation of the absolute value of the instantaneous phase of the example | \[
\sigma_{\text{dp}} = \sqrt{\frac{1}{N_s} \sum_{i=1}^{N_s} \phi_2(i) - \left(\frac{1}{N_s} \sum_{i=1}^{N_s} |\phi(i)| \right)^2}
\]
| The standard deviation of the absolute value of the normalized instantaneous amplitude of the example | \[
\sigma_{\text{aa}} = \sqrt{\frac{1}{N_s} \sum_{i=1}^{N_s} a_n^2(i) - \left(\frac{1}{N_s} \sum_{i=1}^{N_s} |a_n(i)| \right)^2}
\]
| \( a_n(i) = a(i)m_a \) | \( m_a = \frac{1}{N_s} \sum_{i=1}^{N_s} a(i) \) |
| \( N_s \) is the example length |
| The standard deviation of the absolute value of the normalized instantaneous frequency of the example | \[
\sigma_{\text{fn}} = \sqrt{\frac{1}{N_s} \sum_{i=1}^{N_s} f_n^2(i) - \left(\frac{1}{N_s} \sum_{i=1}^{N_s} |f_n(i)| \right)^2}
\]
| \( f_n(i) = f(i)m_a \) | \( m_a = \frac{1}{N_s} \sum_{i=1}^{N_s} f(i) \) |
| \( N_s \) is the example length |
| The kurtosis of the instantaneous phase | \[
K_f = \frac{\sum_{i=1}^{N_s} (f(i) - m_f)^4}{\left(\sum_{i=1}^{N_s} (f(i) - m_f)^2\right)^2}
\]
| \( m_f = \frac{1}{N_s} \sum_{i=1}^{N_s} f(i) \) |
| \( N_s \) is the example length |
| The kurtosis of the instantaneous frequency | \[
K_f = \frac{\sum_{i=1}^{N_s} (f(i) - m_f)^4}{\left(\sum_{i=1}^{N_s} (f(i) - m_f)^2\right)^2}
\]
| \( m_f = \frac{1}{N_s} \sum_{i=1}^{N_s} f(i) \) |
| \( N_s \) is the example length |
| Spectrum symmetry | \[
P_L = \frac{\sum_{i=1}^{N_s} S(i)^2}{P_U} = \frac{\sum_{i=1}^{N_s} S(i)^2}{\sum_{i=1}^{N_s} S(i + f_c + 1)^2}
\]
| \( f_c \) is the sample number corresponding to the carrier frequency |
| \( S(t) \) is the frequency-domain representation of \( s(t) \) |
| The standard deviation of the absolute value of the normalized instantaneous frequency of the example | \[
\sigma_{\text{fn}} = \sqrt{\frac{1}{N_s} \sum_{i=1}^{N_s} f_n^2(i) - \left(\frac{1}{N_s} \sum_{i=1}^{N_s} |f_n(i)| \right)^2}
\]
| \( f_n(i) = f(i)m_a \) | \( m_a = \frac{1}{N_s} \sum_{i=1}^{N_s} f(i) \) |
| \( N_s \) is the example length |
order statistics might be more specific to an AMC setting in which the datasets contain only Phase-Shift Keying (PSK) and Quadrature Amplitude Modulation (QAM) signals [144]. Feature selection in further discussed in Section 7.3.

- The number of bins per histogram. Throughout the results presented, the number of bins per histogram is selected using $n_{\text{bins}} = \lceil \sqrt{k} \rceil$, where $k$ is the number of examples in the test dataset.

- The feature weightings $\alpha_0, \ldots, \alpha_n$. For simplicity, in this work all features are weighted equally, such that

$$\alpha_0 = \ldots = \alpha_{n-1} = \frac{1.0}{n}$$

7.2 Results

7.2.1 Transferability Metrics

When evaluating whether a transferability metric is accurate, the primary consideration is how well the metric reflects or correlates with the performance metric(s) used. In other words, if LEEP and/or LogME can be used to select models for RF TL, LEEP and/or LogME will be correlated with post-transfer top-1 accuracy. To this end, Figures 7.1 and 7.2 shows the LEEP and LogME scores versus the achieved transfer accuracy for each of the synthetic domain adaptation experiments described in Chapter 5, Section 5.1, and Figures 7.3 and 7.4 shows the LEEP and LogME scores versus the achieved transfer accuracy for each of the synthetic sequential learning experiments described in Chapter 6, Section 6.1.

These figures qualitatively show that for all synthetic domain adaptation experiments (Figure 7.1), both LEEP and LogME appear to be positively and linearly correlated with post-
Figure 7.1: The LEEP and LogME scores versus post-transfer top-1 accuracy for the synthetic sweep over SNR, FO, and SNR+FO. The dashed lines represent the linear fits for all target domains.
7.2. Results

(a) LEEP, AMC

(b) LogME, AMC

(c) LEEP, SEI

(d) LogME, SEI

Figure 7.2: The LEEP and LogME scores versus post-transfer top-1 accuracy for the captured domain adaptation experiments with tanh function fits.
Figure 7.3: The LEEP and LogME scores versus post-transfer top-1 accuracy for the synthetic sequential learning experiments colored by target task. The correlation coefficients are averaged over target task.

For the captured domain adaptation experiments, LEEP and LogME seem to be positively correlated with post-transfer accuracy via curved functions such as the Hyperbolic Tangent (tanh) plotted in Figure 7.2. The sequential learning experiment plots shown in Figures 7.3 and 7.4 also show positive correlation generally between LEEP and post-transfer accuracy, but only within the target task for LogME. More specifically, there is a positive correlation between LogME and post-transfer accuracy for all models with the same target task.

While the trends in Figures 7.1-7.4 qualitatively show correlation between LEEP/LogME and post-transfer accuracy, the Pearson correlation coefficient [145] and the weighted $\tau$ [146] are used to quantify this correlation and are specified in the shaded boxes. The Pearson correlation coefficient, or Pearson’s $r$, is a measure of linear correlation between two variables used in a wide variety of works, including the original LEEP paper. However, Pearson’s $r$ makes a number of assumptions about the data, some of which may not be met by this
7.2. Results

(a) LEEP, Model Refinement AMC
(b) LogME, Model Refinement AMC
(c) LEEP, Model Refinement SEI
(d) LogME, Model Refinement SEI
(e) LEEP, Sequential SEI
(f) LogME, Sequential SEI

Figure 7.4: The LEEP and LogME scores versus post-transfer top-1 accuracy for the captured sequential learning experiments with points colored by task. The Pearson’s $r$ correlation coefficient is the averaged over target task.
data. Most notably, Pearson’s $r$ assumes that both variables (LEEP/LogME and post-transfer top-1 accuracy, herein) are normally distributed and have a linear relationship. Alternatively, weighted $\tau$, a weighted version of the Kendall rank correlation coefficient (Kendall $\tau$), is used in the original LogME work. Weighted $\tau$ is a measure of correspondence between pairwise rankings, where higher performing/scoring models receive higher weight, and only assumes the variables (LEEP/LogME and post-transfer top-1 accuracy, herein) are continuous. Both Pearson’s $r$ and weighted $\tau$ have a range of $[-1, 1]$, such that a correlation coefficient of -1 indicates negative correlation (i.e. a high LEEP/LogME score suggests a low post-transfer accuracy), a correlation coefficient of 1 indicates positive correlation (i.e. a high LEEP/LogME score suggests a high post-transfer accuracy), and a correlation coefficient of 0 indicates no correlation (i.e. a high LEEP/LogME score does not suggest either high or low post-transfer accuracy).

The correlation coefficients confirm the qualitative results discussed above. For all domain adaptation experiments, the Pearson’s $r$ correlation is over 0.5 and the average is 0.8270. Likewise, the weighted $\tau$ correlation is always above 0.64, and the average is 0.7938. For the synthetic data sequential learning experiments, using both head re-training and fine tuning, the Pearson’s $r$ correlation averages over 0.7 for all tasks, and the weighted $\tau$ correlation averages over 0.45 for all tasks. Similarly, for the captured data sequential learning experiments, Pearson’s $r$ and weighted $\tau$ average over 0.75 and 0.4, respectively, when using fine tuning. However, when using head re-training, the correlation coefficients are only weakly positive. These results echo those presented in Chapter 6, which showed that fine-tuning vastly outperformed head re-training in a sequential learning setting over all experiments using captured data.

Additionally, Figures 7.5 - 7.7 confirm the LEEP and LogME scores are positively correlated with each other. However, like the correlation between LEEP/LogME and post-transfer
7.2. Results

(a) Domain Adaptation

(b) Sequential Learning

Figure 7.5: The LEEP versus LogME scores for the synthetic domain adaptation (a) and sequential learning (b) experiments. For domain adaptation (a), the dashed lines present the linear fit. For sequential learning (b), the points are colored by target task, and the Pearson’s $r$ correlation coefficient is the average of the correlation within target task.

(a) AMC

(b) SEI

Figure 7.6: The LEEP versus LogME scores for the captured domain adaptation AMC (a) and SEI(b) experiments with linear and tanh curve fits.
accuracy, LEEP and LogME are only positively correlated when the target task or output space remains the same suggesting that LEEP does a better job of considering both domain and task similarity, while LogME is a better judge of domain similarity than task similarity. This result makes sense given that LEEP operates on a network’s logits while LogME operates on the output of a network’s penultimate layer, one layer higher up in the network. More specifically, in a deep NN, the features early layers are considered more general, while the features learned in the later layers are considered more specific [147]. By operating on the logits, LEEP considers the most task-specific features possible within the network, while
the features taken from the penultimate layer for LogME slightly less task-specific.

Together with Figures 7.1-7.4, Figures 7.5 - 7.7 indicates that LEEP and LogME are consistent with both post-transfer top-1 accuracy and with each other. Therefore, in the context of RF TL, LEEP and LogME are good predictors of TL success, so long as the target task remains the same. In other words, LEEP and LogME are useful for selecting a source model selection, a given target dataset. However, LEEP and LogME are not sufficient for identifying good target datasets for a single source model, when the target datasets have different label spaces. These results are also consistent with the results presented in the original LEEP and LogME publications where the metrics were tested in CV and NLP settings, supporting the claim that these metrics are truly modality agnostic. Therefore, other modality agnostic metrics seem likely to perform well in RFML settings as well, and may be examined as follow on work.

7.2.2 Predicting TL Accuracy Using LEEP & LogME

Having confirmed that LEEP and LogME can be used to select models for RF TL, what follows is an approach to not only select models for RF TL, but also to predict the post-transfer top-1 accuracy. The approach is time and resource intensive to initialize, but once initialized, is fast and relatively inexpensive to compute and shows the predictive capabilities of these metrics. Additionally, the cost of initialization can be mitigated somewhat by using only a subset of the available source/target pairs. However, the more source/target pairs used, the better the quality of the transfer accuracy prediction and confidence interval.
CHAPTER 7. METRICS FOR PREDICTING RF TL PERFORMANCE

Approach

Given \( n \) known domains and assuming a single model architecture, to initialize the approach:

1. Run baseline simulations for all \( n \) known domains including pre-training source models on all domains, and using head re-training and/or fine-tuning to transfer each source model to the remaining known domains.

2. Compute LEEP/LogME scores using all pre-trained source models and the remaining known domains.

3. Compute post-transfer top-1 accuracy for all transfer-learned models, constructing datapoints like those displayed in Figures 7.1-7.4.

4. Fit a function of the desired form (i.e. linear, logarithmic, etc.) to the LEEP/LogME scores and post-transfer top-1 accuracies. For example, linear and tanh fits of the forms \( y = \beta_0 x + \beta_1 \) and \( y = \beta_0 \tanh(x) + \beta_1 \) are shown in Figures 7.1-7.4 such that \( x \) is the transferability score and \( y \) is the post-transfer top-1 accuracy.

5. Compute the margin of error by first calculating the mean difference between the true post-transfer top-1 accuracy and the predicted post-transfer top-1 accuracy (using the linear fit), and then multiplying this mean by the appropriate \( z \)-score(s) for the desired confidence interval(s) \([148]\).

Then, during deployment, given a new labeled target dataset:

1. Compute LEEP/LogME scores for all pre-trained source models and new target dataset.

2. Select the pre-trained source model yielding the highest LEEP/LogME score for TL.
3. Use the fitted function to estimate post-transfer accuracy, and add/subtract the margin of error to construct the confidence interval.

Optionally, after transferring to the new labeled target dataset, add this dataset to the list of known domains, and update the fit function and margin of error, as needed.

**Accuracy of the Proposed Approach**

The error in the predicted post-transfer accuracy using the proposed method is shown in Figures 7.8 - 7.11. These plots show that not only are LEEP/LogME correlated with post-transfer top-1 accuracy, as was shown in Figures 7.1-7.4, and highly correlated with one another, as was shown in Figures 7.5 - 7.7, but the error in the predicted post-transfer top-1 accuracy using a linear fit to the LEEP and LogME scores respectively is also highly correlated. More specifically, when the proposed method constructed using LEEP predicts a lower/higher post-transfer accuracy than ground truth, the proposed method constructed using LogME will do the same with the frequencies shown in Table 7.2. This indicates that these scores could be combined to create a more robust transferability metric and more robust post-transfer accuracy prediction with relative ease, which is left for future work.

**7.2.3 Dataset Similarity**

One of the primary goals of the proposed dataset similarity metric is to intuitively quantify similarity between raw RF datasets. While Figures 7.12-7.13 show that similarity increases as the intersection or overlap in SNR, FO, and modulation schemes in the source/target datasets increases using synthetic data, there is still room for improvement. Of note, Figure 7.12 shows that the metric is sensitive to relatively small changes in SNR and is symmetric with regards to FO (i.e. datasets with equal amounts of FO in the positive and negative
Figure 7.8: The error in the predicted post-transfer accuracy using a linear fit to the LEEP scores (x-axis) and LogME scores (y-axis) for the synthetic domain adaptation experiments.
7.2. Results

(a) Synthetic Sequential AMC

(b) Synthetic Model Refinement AMC

Figure 7.9: The error in the predicted post-transfer accuracy using a linear fit to the LEEP scores (x-axis) and LogME scores (y-axis) for the synthetic sequential learning experiments.

(a) AMC

(b) SEI

Figure 7.10: The error in the predicted post-transfer accuracy using a tanh fit to the LEEP scores (x-axis) and LogME scores (y-axis) for the captured domain adaptation experiments.
Chapter 7. Metrics for Predicting RF TL Performance

Figure 7.11: The error in the predicted post-transfer accuracy using a tanh fit to the LEEP scores (x-axis) and LogME scores (y-axis) for the captured sequential learning experiments.

(a) Model Refinement, AMC  
(b) Model Refinement, SEI  
(c) Sequential, SEI
7.2. Results

Figure 7.12: Similarity across the synthetic datasets with varying (a) SNR, (b) FO, and (c) SNR and FO, with other metadata parameters held constant.
CHAPTER 7. METRICS FOR PREDICTING RF TL PERFORMANCE

Figure 7.13: Similarity across the synthetic datasets containing only linear, only frequency-shifted, and only analog modulation schemes, and a mixture of modulation types (a) and the synthetic datasets with a single modulation scheme added/removed (b). All other metadata parameters are held constant.

Figure 7.14: Similarity across the captured datasets with varying domains (a) and tasks (b) for the AMC use-case.
7.2. RESULTS

<table>
<thead>
<tr>
<th>Synthetic Experiments</th>
<th>Head Re-Training</th>
<th>Model Fine-Tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNR Sweep, AMC</td>
<td>0.6175</td>
<td>0.7496</td>
</tr>
<tr>
<td>FO Sweep, AMC</td>
<td>0.7856</td>
<td>0.8803</td>
</tr>
<tr>
<td>SNR + FO Sweep, AMC</td>
<td>0.7258</td>
<td>0.7468</td>
</tr>
<tr>
<td>Sequential, AMC</td>
<td>0.6000</td>
<td>0.6000</td>
</tr>
<tr>
<td>Model Refinement, AMC</td>
<td>0.8056</td>
<td>0.8194</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Captured Experiments</th>
<th>Head Re-Training</th>
<th>Model Fine-Tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain Adaptation, AMC</td>
<td>0.7006</td>
<td>0.7130</td>
</tr>
<tr>
<td>Domain Adaptation, SEI</td>
<td>0.8488</td>
<td>0.8086</td>
</tr>
<tr>
<td>Sequential, AMC</td>
<td>0.7500</td>
<td>0.7500</td>
</tr>
<tr>
<td>Sequential, SEI</td>
<td>0.6875</td>
<td>0.8125</td>
</tr>
<tr>
<td>Model Refinement, SEI</td>
<td>0.9375</td>
<td>0.7297</td>
</tr>
</tbody>
</table>

Table 7.2: The frequency with which the proposed method constructed using LEEP and LogME agree in over/under predicting post-transfer accuracy.

direction – for example, [-10%, -5%] of sample rate and [5%, 10%] of sample rate – are highly similar). Additionally, the relative changes in SNR across captured data domains discussed previously in Chapter 5 seems to also be affecting the ability to quantify similarity between these datasets. These effects are likely a result of the instantaneous time-domain features used to provide baseline results for an AMC setting, and could likely be mitigated using a different feature set, a topic further discussed in Section 7.3.

Additionally, the proposed dataset similarity metric aims to quantify source/target similarity through datasets similarity. Given that the results of the previous chapters has shown that TL performance correlates with source/target similarity in TL performance, ideally TL performance would also correlate with the proposed dataset similarity metric. While Figure 7.15 shows that similar synthetic source/target datasets, as quantified by the proposed metric, result in successful RF TL by showing that the metric positively correlates with improved performance of the TL models over baseline models, Figure 7.16 shows that captured dataset similarity is only weakly correlated with improved TL performance. These results
Figure 7.15: The difference between the accuracy of TL models versus baseline models, in decimal format, as a function of the proposed metric for the synthetic dataset. When the difference is above zero, as shown by the solid black line, the TL model outperforms the baseline model.

Figure 7.16: The difference between the accuracy of TL models versus baseline models, in decimal format, as a function of the proposed metric for the captured data AMC use-case. When the difference is above zero, as shown by the solid black line, the TL model outperforms the baseline model.
### 7.2. Results

<table>
<thead>
<tr>
<th></th>
<th>Full-Data Baseline</th>
<th>Limited-Data Baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Synthetic Experiments</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNR Sweep</td>
<td>0.5634</td>
<td>0.5799</td>
</tr>
<tr>
<td>FO Sweep</td>
<td>0.5087</td>
<td>0.4333</td>
</tr>
<tr>
<td>SNR + FO Sweep</td>
<td>0.5243</td>
<td>0.5364</td>
</tr>
<tr>
<td>Modulation Types</td>
<td>0.2440</td>
<td>0.2860</td>
</tr>
<tr>
<td>Model Refinement</td>
<td>0.2293</td>
<td>0.2494</td>
</tr>
<tr>
<td><strong>Captured Experiments</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Domain Adaptation</td>
<td>0.1802</td>
<td>0.0903</td>
</tr>
<tr>
<td>Sequential</td>
<td>0.2987</td>
<td>0.4453</td>
</tr>
</tbody>
</table>

Table 7.3: The Pearson’s $r$ correlation coefficient between difference between the accuracy of TL models versus baseline models and the proposed dataset similarity metric for all AMC experiments.

are quantified using the Pearson’s $r$ correlation coefficient in Table 7.3.

Despite positive correlations between synthetic dataset similarity and TL performance, a higher/lower expert feature-based similarity score does not directly infer that TL provides performance benefits over training from random initialization. More specifically, because RFML approaches eschew expert-defined features in favor of raw RF input, how well the metric correlates with TL performance is dependent on the expert features chosen and whether or not they correlate with the features naively learned by the ML/DL model. Even when dataset similarity is high, if sufficient data is available, training from random initialization on the target dataset is preferred to using TL, as shown in Figures 7.15b and 7.16b.

Given that a high/low similarity score does not directly infer TL performance, the question becomes: “Is this metric useful in helping to determine whether it is more beneficial to use TL approaches or train from random initialization?” First, causality is not expected because dataset similarity is only one facet of TL performance. Instead, the proposed dataset similarity should be viewed as a tool for rank prioritizing the raw RF datasets most likely to be fruitful for TL in an RFML setting. However, in the case where the size of the target
Figure 7.17: The difference between the accuracy of TL models versus baseline models, in decimal format, as a function of the proposed metric for the Synthetic Model Refinement AMC experiment. When the difference is above zero, as shown by the solid black line, the TL model outperforms the baseline model.

Dataset is limited (as shown in Figure 7.15a, for some use-cases there is a similarity threshold at which all models perform better using TL over training from random initialization. For example, Figure 7.17 isolates the points in 7.15a from Synthetic Sequential Experiment 2, and shows that such a threshold is present when similarity is greater than 0.63.

Finally, Figure 7.18 examines the impact of dataset size and the number of histogram bins on the proposed dataset similarity metric. Throughout each of the experiments mentioned above, the proposed dataset similarity metric was computed using 500 examples per output class and setting $n_{\text{bins}} = \lfloor \sqrt{500 \cdot m} \rfloor$. Because $n$ features are computed for each example in each dataset, the proposed dataset similarity metric operates on histograms of those features, and the number of histogram bins is selected as a function of dataset size, we expect that the metric is largely invariant to dataset size, so long as a sufficient number of examples are available to characterize the dataset. This “sufficient” number of examples
7.2. Results

Figure 7.18: Dataset similarity as a function of dataset size and the resultant number of bins per histogram used in each $\chi^2$ test for the first column of source/target dataset pairs shown in Figure 7.13b, averaged over 10 iterations.

is likely subject to the diversity of the datasets, with more diverse datasets requiring more examples to characterize each dataset. Figure 7.18 plots the similarity between the $\Phi_{S1}$ and $\Phi_{S2}$, $\ldots$, $\Phi_S$ (the first column of source/target pairs in Figure 7.13b) as a function of dataset size (top axis) and the number of histogram bins (bottom axis). Results show that when the datasets are too small (i.e. 100 examples per dataset), similarity is artificially high for all dataset pairs. As dataset size increases, similarity steadily decreases. However, similarity is relatively consistent between source/target pairs with as few as 900 examples per dataset and 30 bins per histogram, meaning the similarity between $\Phi_{S1}$ and $\Phi_{S2}$ is almost always higher than the similarity between $\Phi_{S1}$ and $\Phi_{S3}$. While larger datasets with more bins per histogram are generally advantageous when computing the proposed similarity metric, as the separation between dataset pairs increases, meaningful conclusions can be drawn from much smaller datasets.
7.3 Discussion

This chapter has focused on metrics and methods for source model selection without additional training and predicting RF TL behavior without knowledge of the source/target metadata and labels. First, two existing modality agnostic transferability metrics, LEEP and LogME have been evaluated in the context of RF TL. Results have shown both qualitatively and quantitatively that LEEP and LogME are positively correlated with post-transfer top-1 accuracy, assuming a constant target task, for both synthetic and captured datasets and AMC and SEI use-cases, concluding that these metrics are useful for source model selection in an RFML setting. Additionally, this chapter has presented an approach for using any transferability metric to predict post-transfer top-1 accuracy within a confidence interval.

This chapter has also presented a novel RF-specific dataset similarity metric to quantify RF domain and task similarity through expert-defined feature distributions. The proposed metric is intuitive, flexible, computationally efficient, invariant to dataset size, and does not require labels, but does require some expert knowledge to select features applicable to the use-case-of-interest. For the proof of concept, 7 instantaneous time-domain features were used to measure dataset similarity for an AMC use-case, which was tested across both synthetic and captured data. Initial results presented using the synthetic data showed that the metric is capable of intuitively quantifying similarity, but is sensitive to changes in SNR, impacting the quality of the results on captured data. Nevertheless, the proposed metric is an additional tool for rank prioritizing and categorizing raw RF datasets for RF TL, and could be further refined using different feature sets in future work.

More specifically, in order for the proposed dataset similarity metric to be most useful across different RF use-cases, expert-defined feature selection is of the utmost importance, and
identifying better feature selection criteria and/or combining the proposed dataset similarity metric with formal feature selection methods would increase the utility of the proposed metric across tasks and domains. It should be noted that the problem of identifying the “optimal” features for quantifying raw RF dataset similarity is ill-defined. That is, the notion of RF dataset similarity likely changes depending on the use-case-of-interest, and no ground-truth similarity score exists. However, at minimum, selecting better feature sets for quantifying raw RF dataset similarity should consider both the relevance of the features to the use-case-of-interest, as well as maximizing the orthogonality between features to minimize redundancy. Future feature selection methods could tackle these criteria separately or in tandem.

Starting with the concept of feature relevance, two approaches can be taken: using expert knowledge to identify relevant feature from which to perform feature selection, or creating some notion of “ground truth” similarity (using post-transfer accuracy, for example) that can be used to selecting features relevant from a wide pool of candidate features. When considering feature orthogonality, possible approaches include computing the correlation coefficient between candidate features for a single dataset [149], using distance metrics such as the KL divergence [150], JS divergence [151], or Bhattacharyya distance [152] to select for dissimilarity between feature distributions for individual datasets and/or between datasets. If considering feature relevance and orthogonality in tandem and assuming some user-defined notion of “ground truth” similarity, possible feature selection methods also include Principal Feature Analysis (PFA) [153] and genetic algorithms [154].
Chapter 8

Conclusion

TL is a pervasive technology in CV and NLP, yielding exponential performance improvements by leveraging prior knowledge gained from data with different distributions. However, while recent works seek to mature ML and DL techniques in applications related to wireless communications, few have demonstrated the use of TL techniques for yielding performance gains, improved generalization, or to address concerns of training data costs. Designing TL algorithms for the RFML space first requires a fundamental understanding of how the RF domain and RF tasks impact learned behavior and inhibit or facilitate transfer. Even for RFML works that have successfully used TL, such limits in understanding may hinder further performance improvements that might be yielded from TL techniques. Additionally, these limitations in understanding also obscure insights into long-term model behavior during deployment, which has long been a criticism of RFML and prevented commercial support and deployment [7].

This dissertation has sought to bridge this gap, providing a fundamental understanding of how TL techniques perform under various RF conditions. More specifically, Chapter 1 presented the motivations for using RF TL, and identified the following research questions for this work:

1. What are the underlying system parameters affecting the effectiveness of transferring learned behaviors?
2. What are general guidelines/best practices for RF TL?

3. What are some of the unique challenges and limitations of RF TL?

4. Can TL performance reasonably be predicted?

In Chapters 2 and 3, an RF domain-specific TL taxonomy and survey of RFML and RF TL research was presented to clearly define the problem space, provide sufficient background, distinguish RF TL from TL in other modalities, and further motivate RF TL, partially addressing research question 3. To more fully address question 3 and tackle questions 1 and 2, RF TL performance was systematically evaluated across changes in channel type, SNR, CF, FO, Tx/Rx hardware, and modulation type for AMC and SEI use-cases across two different NN architectures and both synthetic and captured data in Chapters 5 and 6, using the experimental framework presented in Chapter 4. Finally, moving towards the more practical elements of using RF TL in real-world settings, Chapter 7 studied existing modality agnostic transferability metrics, LEEP and LogME, in an RFML setting, introduced a novel RF dataset similarity metric, and presented a method for using any transferability metric to predict post-transfer accuracy, addressing research question 4. Through this exhaustive study, a number of guidelines have been identified for when and how to use RF TL successfully, and many additional directions for future research have been identified, each the subject of the following sections.

8.1 Generalized Guidelines for RF TL

Over the course of this dissertation, 5118 models were trained over 81 different domains and 43 different tasks, systematically varying SNR, FO, CF, modulation schemes, Tx hardware, Rx hardware, and channel. Given the careful curation of the signal parameters contained
within each data subset, the breadth of signal types and parameters observed, and experimentation over both synthetic and real data, AMC and SEI use-cases, and two different NN architectures, generalized conclusions can be drawn regarding how the domain and task impact TL performance.

Most important is understanding when RF TL is useful. In this work, two baselines were compared to TL performance: a limited-data baseline, trained with a dataset equal to the size of the target dataset, and a full-data baseline, trained with a dataset equal to the size of the source (pre-training) dataset. In most cases, when a full-size labeled dataset is available for the target domain/task, there is no accuracy benefit to using TL over training from random initialization. However, TL is more time and computationally efficient than training from random initialization. So, if time and/or computational resources are limited, and a full-size training dataset and suitable source model are available, using TL may be of benefit. One caveat here is that further research is needed to investigate TL performance across changes in source/target dataset size, training hyperparameters, etc.

When only limited amounts of target training data are available, the results presented herein suggest that TL almost always achieves better performance over training from random initialization, in terms of accuracy, time, and computational complexity, but more research is needed to confirm this trend. In the experiments over synthetic datasets, TL only provided top-1 accuracy improvements over the limited-data baselines when the source and target domains and tasks are “similar.” However, in the captured dataset experiments, TL outperforms the limited-data baseline, in terms of accuracy, in most settings. Given that the captured dataset is more representative of what would be encountered by a real-world system, as well as the time and computational complexity benefits of using TL versus training from random initialization, these early results support the use of TL in most settings where representative training data is limited.
Assuming limited amounts of training data, and given that TL is beneficial over training from random initialization most of the time, there are some more fine grained points of RF TL performance to consider. First, when performing domain adaptation, in addition to domain similarity, domain difficulty is also important. Additional research is needed to understand the origins of domain difficulty and discussed further in the next subsection, but when one domain is more difficult than the other, TL performance is better when transferring from the harder domain to the easier domain. What is well understood from the results presented in this dissertation is that the channel, and subsequent impact on SNR, is a contributing factor to domain difficulty, and must be considered when attempting to perform RF TL, as it has a significant impact on performance. Meanwhile, FO and Tx/Rx hardware impact RF TL performance to a much lesser degree. In other words, environment adaptation and environment platform co-adaptation are far more challenging than platform adaption.

When performing sequential learning over changes in task, it is beneficial for there to be some overlap between the source and target task, such that the intersection between the source task outputs and target task outputs is non-empty, the source task is a subset of the target task, or vice versa. The best sequential learning performance is achieved when the target task is a subset of the source task. For example, when performing model refinement, results showed better performance when removing output classes than when adding output classes. This guideline can be attributed to the fact that (1) the target task gets easier as output classes are removed and (2) the model does not need to do any significant feature learning from the target data. Instead, the task is simplified, and any unnecessary features learned previously can be eliminated.

Finally, while further research is needed to better understand how to best perform RF TL, the results presented herein have shown the relative benefits of using head re-training versus fine-tuning. More specifically, when performing domain adaptation, head-retraining generally
performed as well, if not better than, fine-tuning, and when performing sequential learning across changes in task model fine-tuning vastly outperformed head re-training. Depending on the size of the model and the computational resources available, model fine-tuning may be quite a bit more expensive than head re-training, both computationally and in terms of training time. For the CNN and CLDNN model architectures used in this work, using head re-training versus fine-tuning resulted in an overall reduction of 99.98% and 99.27% in trainable parameters, respectively.

8.2 Future Work

As discussed throughout this dissertation, RF TL largely remains an open area of research. However, the areas most relevant to the work performed herein can be categorized into continued RF TL analysis and algorithmic development, each discussed in the following subsections.

8.2.1 Additional RF TL Analysis

Continuing and extending the analysis conducted Chapters 5 and 6 will provide a more thorough understanding of RF TL behavior and performance across a wider range of use-cases and deployment settings. Additional suggested directions for experimentation include:

- Analysis of multi-task learning behavior using synthetic and captured data. While work in [109] showed a proof-of-concept approach for performing multi-task AMC and SEI using a combination DenseNet and Transformer network architecture, and showed improved performance over single-task models, the approach has yet to be evaluated on captured data or across changes in domain and task.
• Analyses of TL performance for other RFML use-cases, such as signal detection and spectrum anomaly detection, and additional model architectures, such as Transformer-based or generative network architectures.

• Analysis of RF TL performance across changes in both domain and task or simultaneous domain adaptation and sequential learning.

• An analysis of RF TL techniques for transferring between use-cases. For example, examining transfer between AMC and SEI use-cases.

• Analysis of RF TL performance across synthetic, augmented, and captured datasets. Work in [12] has shown that training RFML models on synthetic data alone does not yield sufficient performance on captured, real-world data for deployment. However, by intelligently augmenting the data to match the SNR, FO, and sample rate mismatch to the deployed environment can greatly improve performance on the captured data. If small amounts of labeled captured data are available for training, TL provides an approach to further improve the performance of models pre-trained on synthetic or augmented data in real-world settings.

Next, the results in Chapter 5 showed that TL performance was not solely predicted by metadata parameters such as CF, Rx ID, or channel, but was correlated with baseline performance or domain difficulty. This has raised the question: What dictates domain difficulty? While SNR is a clearly contributing factor to domain difficulty, it does not seem to be the only factor. To address this question more fully, experiments including more CF, Rx ID, and channel variations and varying additional parameters-of-interest, such as fading/multipath channel environments, sample rate, and temperature would provide greater context and more data points from which to derive performance trends. Additionally, cleaner, more controlled captured datasets, perhaps collected in an anechoic chamber, and the use of data
augmentation would provide more granular variation in domain, yielding smoother trends. Additionally, as mentioned in Chapter 7, several new transferability metrics were developed concurrently with this work. Given that this work supports the claim that LEEP and LogME are modality agnostic, it seems likely that additional transferability metrics that are also modality agnostic by design will also follow this trend. However, the experiments of Chapter 7 could be replicated using these alternative metrics, which include OTCE [127], JC-NCE [128], TransRate [129], and GBC [130], to identify if they are also suitable for use in the context of RFML and if they might outperform LEEP and LogME.

Furthermore, this dissertation has focused on how changes in the RF domain and task impact TL performance, keeping the TL methods fixed. Future research should address the counter side of this work by examining RF TL performance as a function of various TL hyperparameters and training schemes, including

- Using unsupervised, self-supervised, semi-supervised, or weakly supervised pre-training methods,
- Varying learning rate and/or using learning rate schedulers,
- Varying source and target dataset size,
- Varying the number of layers frozen and/or fine-tuned, and
- Using fine-tuning methods such as chain thaw [155] or gradual/scheduled unfreezing [156, 157],

while generalizing over the source and target domains and tasks.

Finally, as RFML research branches out into unsupervised and self-supervised techniques, the RF-specific TL taxonomy presented in Chapter 3 will need to be expanded to include
transductive learning techniques like those presented in the general taxonomy of [79].

8.2.2 Algorithmic Development

Research into better performing RF TL that is robust across a wider variety of real-world settings is perhaps the most apparent direction for future work. Most readily, parallels can be drawn between RFML and other modalities in which TL has been employed successfully, in an effort to identify existing methods that can be borrowed. For example, many of the DL architectures used in RFML are CNN-based, like in CV, yielding a large selection of TL methods from which to work from. Alternatively, the sequential nature of raw RF data is more akin to that used in text or audio modeling, yielding additional NLP TL methods. However, borrowing such approaches yields no guarantee of success. Just as TL in text domains is characteristically different from TL in visual domains [158], thereby requiring different approaches and techniques, it is possible that wholly new TL algorithms will need to be developed for the RFML space.

One such direction is seeking to develop methods and architectures for learning universal feature representations. The most likely solution to this problem is the development of an RF foundation model, trained in an unsupervised or semi-supervised manner, much like BERT [159], GPT [160], or wav2vec [161]. The development of foundation models and universal feature representations generally requires large amounts of data, though unlabeled, large architectures, and large amounts of compute, but once trained can sometimes be transferred to any future downstream domains and tasks, with little-to-no additional training.

Finally, while Chapter 7 has examined methods and metrics for source model selection and predicting post-transfer accuracy, future work is needed to develop methods and metrics for initiating TL. While one could regularly re-train or fine-tune at fixed time intervals,
making the assumption that the RF domain/task is changing at a rate slower than the re-training/fine-tuning interval, a more optimal solution would be to identify, in real-time, when the current inputs are no longer representative of the training data and/or the deployed model has degraded in performance. Promising methods include those used to identify dataset drift or covariance shift [162], methods for detecting out-of-distribution examples [163, 164], as well as uncertainty quantification methods such as temperature scaling [165] or Bayesian approximation [166]. Such methods are not only required for initiating TL, but also help provide user-assured performance, and are important for ruggedizing the decision chain against spoofing and other adversarial techniques.
Bibliography


163


[75] Z. C. Lipton, “The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery,” *Queue*, vol. 16, p. 31–57, jun 2018.


