Intelligent Anti-jamming based on Deep Reinforcement Learning and Transfer Learning

Siavash Barqi Janiar

A thesis submitted to
the Faculty of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree of
Master of Applied Science

Graduate Program in
Electrical Engineering and Computer Science
York University
Toronto, Ontario

Abstract

One of the security issues in a wireless network is jamming attacks, where the jammer causes congestion and significant decrement in the network throughput by obstructing channels and disrupting user signals. In this thesis, we first develop a deep reinforcement learning (DRL) model to confront the jammer. However, training a DRL model from scratch may take a long time. We further propose a transfer learning (TL) approach to enable the DRL agent to learn fast in dynamic wireless networks to confront jamming attacks effectively. To make our proposed TL method adaptive to different network environments, we propose a novel method to quantitatively measure the difference between the source and target domains, using an integrated feature extractor. Afterward, based on the measured difference, we demonstrate how it can help choosing an efficient setting for the TL model leading to a fast and energy-efficient learning. We also show that the proposed TL method can effectively reduce the training time for the DRL model and outperforms other existing TL methods.

Acknowledgement

I am pleased to acknowledge that a conference version of this thesis is accepted at IEEE MetaComm Conference with 35% acceptance rate. A journal version of this thesis also will be submitted in IEEE Transactions on Vehicular Technology.

I would like to express my deepest gratitude to my supervisor, Prof. Ping Wang, for her guidance and support throughout my thesis project. Her expertise, patience, and dedication have been invaluable to my academic and personal growth. I am grateful for her insightful feedback, challenging questions, and unwavering encouragement, which have helped me to refine my research and writing skills.

I would also like to thank my supervisory committee member and examiner, Prof. Ruth Urner and Prof. Jinjun Shan, respectively, for their time and effort in reviewing my thesis, as well as the faculty members and staff of EECS for their contributions to my education. Their passion for teaching and research has inspired me to pursue academic excellence and intellectual curiosity.

Finally, I would like to thank my family and friends for their love, support, and encouragement. Their unwavering belief in me has been a source of strength and motivation, and I am deeply grateful for their presence in my life.

Thank you all for making this thesis possible.

Contents

Abstract ii

Acknowledgement iii

Contents iv

List of Tables v

List of Figures vi

Abbreviations vii

1 Introduction 1

1.1 Contributions 2

1.2 Organization of the Thesis 3

1.3 Related Publication 3

2 Related Work 4

3 System Model 7

4 The Proposed Anti-Jamming Method 9

4.1 Deep-Reinforcement Learning 9

4.2 IFE-based Transfer Learning 11

4.3 Feature Relevance Explanation 14

5 Numerical Results 17

6 Conclusion 26

6.1 Summary of the Thesis 26

6.2 Future Directions 26

List of Tables

5.1 Simulation hyperparameters 18
5.2 Training parameters of different models 21
5.3 Relative entropy values obtained from the simulation. 23

List of Figures

3.1 Studied System Model 8

4.1 The Proposed Methodology 9
4.2 Part of a sampled DT from the random forest model. 16

5.1 The performance of different DRL models and Random. Since Random does not have any learning parameters, it has not scaled in this figure. . . 20
5.2 The importance of the features. 22
5.3 IFE based TL models convergence rates for different scenarios 24
5.4 Different ML models convergence rates for different scenarios 25

Abbreviations

ML Machine Learning

DNN Deep Neural Network

DRL Deep-Reinforcement Learning

TL Transfer Learning

IFE Integrated Feature Extractor

CNN Convolutional Neural Network

RNN Recurrent Neural Network

AP Access Point

UAV Unmanned Aerial Vehicle

RCNN Recurrent Convolutional Neural Network

RCL Recurrent Convolutional Layers

WI Weight Initialization

IoT Internet of Things

FRE Feature Relevance Explanation

LSTM Long Short-Term Memory

XAI eXplainable Artificial Intelligence

GDPR General Data Protection Regulation

DT Decision Tree

Chapter 1

Introduction

As wireless networks grow, the technology of wireless network attackers grows and becomes more complicated as well. [1] is an example of a work trying to design a jamming attacker as efficient as possible. This study helps us understand how proficient, efficient, and intelligent a jamming attacker can be designed. Hence, confronting them necessitates using smart and efficient methods.

It has been shown in [2] and [3] that one of the efficient ways to confront jamming attackers is to predict their behaviour using machine learning (ML). In a wireless network with the presence of jammers, the network environment is dynamically changing due to the unpredictive behaviours of jammers, thus gathering data and making a data set available for ML algorithms can be difficult or even impossible. This makes most of ML approaches such as supervised learning and unsupervised learning impractical as for those a dataset is needed before the start of the learning process. However, a group of ML techniques and algorithms do the learning process while there is no data set available. This branch of ML is called *deep-reinforcement learning* (DRL). The learning process in a DRL algorithm happens in an iterative way without requiring any prior information from the environment, which matches the dynamic nature of wireless networks.

1.1 Contributions

However, training a DRL model may take a considerable amount of time and energy. Every time when the network environment changes (e.g., the jamming pattern changes), the DRL model needs to be re-trained to adapt to the new environment. Recently in machine learning, a branch of techniques facilitating the learning process have been emerged named Transfer Learning (TL) [4, 5]. Instead of learning from scratch, the knowledge that is learned in the first place (source domain), if possible, can be transferred to the new place, which is trying to do the same learning process (target domain). In this way, the learning process in the target domain can be accelerated and the time and energy spent on learning can be largely reduced. In our study, we propose a TL approach to speed up the training process of the DRL model when the network environment changes. Different settings are needed for the proposed adaptive TL algorithm depending on how different the source and target environments are in a TL model. To realize that, we devised a method to utilize *integrated feature extractor* (IFE) to measure the differences between source and target domains using *relative entropy* [6]. This method allows us to choose an optimum setting for the TL model in the target domain and achieve the highest possible time and energy efficiency in the TL model.

The following lists the main contributions of this thesis:

- We deploy a more suitable *deep-neural network* (DNN) architecture for the DRL model to address the jamming attack issue in wireless networks, which can largely reduce the computational complexity as compared with the conventional methods. More specifically, unlike most of the works that employed *convolutional neural network* (CNN), we suggest *recurrent neural networks* (RNNs) and illustrate the superiority of the proposed DNN architecture by comparing their performance under the same settings.

- We propose an IFE-based TL approach to enable the DRL agent to learn fast in the dynamic wireless networks to confront jamming attacks effectively. The proposed

method is automatically adaptive in a dynamic environment with different jamming settings. To the best of our knowledge, we are the first to propose employing IFE-based TL approach to address the jamming attacks in wireless networks.

- We propose a novel method to quantitatively measure the difference between the source and target domains. Such measurement results will be used to determine the IFE-baesd TL model settings. This makes our model adaptable to different jamming strategies and avoids unnecessary computations. We compare our proposed method with different TL models and demonstrate the superiority of our method over the other TL methods.

1.2 Organization of the Thesis

In the following, we first review the related works in Chapter 2. Then, we describe the system model in detail in Chapter 3. The proposed methods are explained in Chapter 4. Afterward, the numerical results are presented in Chapter 5. Finally, Chapter 6 summarizes the thesis.

1.3 Related Publication

S. B. Janiar, P. Wang, ”A transfer learning approach based onintegrated feature extractor for anti-jamming inwireless networks,” in Proceedings of *IEEE International Conference on Metaverse Computing, Networking and Applications (MetaCom)*, Kyoto, Japan, 26-28 June 2023.

Chapter 2

Related Work

An introduction to radio jamming can be found in [7]. A jammer is a kind of attacker who attempts to disrupt the communication network performance by sending jamming signals which interfere with the network signals. There are different jamming techniques while on the other hand, there are ways to confront them as well. [8] surveys some existing jamming and anti-jamming techniques. In this survey, *channel hoping* has been introduced as one of the ways to confront the jamming attacker [9, 10, 11, 12, 13, 14]. More specifically, [9] proposed using *reactive* and *proactive* channel hopping to confront the jamming attacker. [10] studied anti-jamming techniques considering other wireless network layers, combined techniques in each layer together with channel hopping, and came up with an anti-jamming algorithm. [11] proposed channel hopping following a pseudo-random sequence which is known only by the access point (AP) so that the jammer will not be able to predict the next communicating channel. [12] implemented the pseudo-random channel hoping method using transceivers and real data. [13] introduced an *adaptive channel hopping* method. The idea of adaptive channel hopping is to consider the channel that has been determined having an acceptable performance as the main channel and every now and then choose other channels so that they would be estimated by the jammer to be jammed avoiding the main channel being jammed. If the main channel is jammed, another channel showing the best performance will be selected as the main channel. The authors

in [14] proposed a code-controlled message-driven frequency hopping scheme that integrates the channel selecting process with channel coding to detect active channels more accurately.

ML and DRL have helped address many existing challenges in communication systems in recent decades. [15] reviews some of the DRL applications in different wireless network areas. One example of DRL usage in wireless network security can be found in [16], which focuses on unmanned aerial vehicle (UAV) security. [17] studies the use of DRL in transmit power control in wireless networks. [18] reviews some of the applications of DRL in wireless network routing. Authors in [19] used DRL to determine the file popularity patterns in caching. Also, [20] utilizes DRL to solve proactive caching problem by dynamically caching or removing contents before requested by the user. A DRL-based approach is used in [21] to detect spoofing attacks in wireless networks. An application of DRL in wireless mesh network communication flow control can be found in [22].

Channel hopping is facilitated with DRL in some recent works such as [23, 24, 3, 2]. Authors of [23] studied the use of DRL and channel hoping in spectrum access to fairly allocate available frequency channels to the active users in a wireless network. Almost the same resource allocation problem is addressed in [25] considering an IoT network. Another resource allocation study is [26] where the studied wireless network is heterogeneous making it a partially observable Markov decision process (POMDP) problem. An extension of the same problem has been studied in [27] where the user signals are prioritized. [24] studied DRL-based channel hoping for cognitive radar facing a jamming attacker. Another anti-jamming work using DRL is [28] where the DRL has been employed for secondary users in a cognitive radio network.

[29] studied an anti-jamming technique in a POMDP problem using long short-term memory (LSTM) and RNN. The authors in [3] however, employed a CNN to solve the jamming problem in a wireless network. The authors in [2] considered the same problem, and as their proposed method, they combined CNN and RNN and came up with a *recurrent convolutional neural network* (RCNN) using *recurrent convolutional layers* (RCLs)

[30] to reduce the number of parameters in an ordinary CNN. However, the number of parameters for the proposed CNNs is yet significantly large.

As an emerging ML technique, TL has been employed in ML since the last decade to improve learning speed. In recent years, TL has also been employed in many communication systems. More specifically, [31, 32, 33] studied the use of TL in anti-jamming wireless networks. [31] studied the jamming attacks in in-body sensors in a wireless body area network using RL and TL. The authors in [32] proposed a way to change the exploration rate for the model in the target domain using relative entropy, which is calculated based on the different jamming probability distributions. However, it is not practical to assume that users have access to the jammer's jamming probability distributions. The idea of [33] is to confront an intelligent jammer using the DRL model used for jamming decisions. So, both users and the jammer use the same DRL model and have access to the same DNN. However, the jammer and users have different objectives, and the answer to the critical question of whether they can follow the same policy to fulfil their objectives by using the same model has not been discussed in the paper. [32, 33] used a *weight initialization* (WI) transfer learning [34] which means the weights of the trained DNN in the source domain are used as initial values for the weights of the DNN in the target domain. One of the limitations of using WI is that it does not reduce the number of parameters. Even though it makes the convergence of the model faster but does not reduce the model's complexity. On the contrary, IFE transfer learning makes the target model significantly simpler if applied and implemented correctly.

Chapter 3

System Model

As depicted in Fig. 3.1, the studied wireless network consists of a number of radio access points (AP), each serving several wireless users, e.g., internet of things (IoT) devices or mobile sensing robots, in a particular region with a frequency hopping based system. All the APs are connected via a backhaul network. There are N channels available at each AP. The time is slotted. At each time slot, there is one active user associated with an AP, and the AP chooses one channel to communicate with the active user. The user signal takes τ time slots to be transmitted. The jammer tries to disrupt the user signal by transmitting the jamming signals on the same channel. Each AP is equipped with a DRL-based agent, which makes decisions (i.e., which channel is selected for transmission) based on its observations from the wireless network.

At any time slot while the user signal is being transmitted, if the jammer succeeds in disrupting the signal by jamming the same channel, the user signal transmission is considered disrupted. Otherwise, the transmission is considered a successful one. As a result, the throughput of the wireless network is defined as the ratio of successful transmissions to all the transmission attempts. Hence, the objective of the wireless network is to maximize the throughput of the network.

There are different types of jammers [8], and in this thesis we consider two types of jammers:

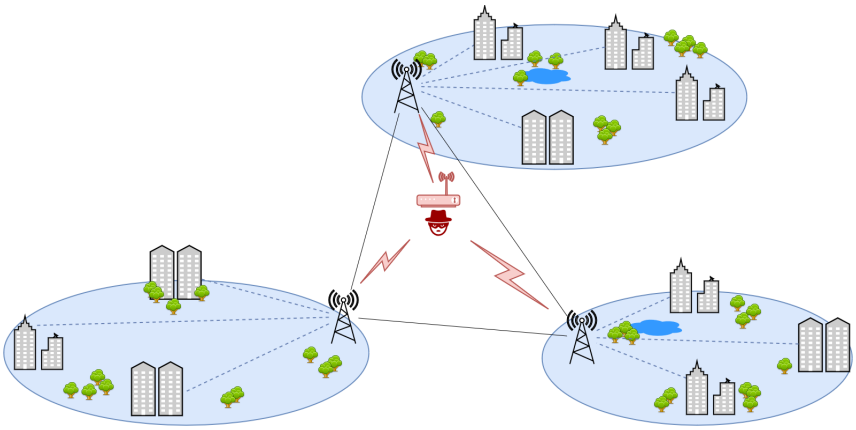


Figure 3.1: Studied System Model

- Sweep jammer: The jammer sweeps all the frequency channels cyclically according to a fixed order to find the chance of disrupting the user signal.
- Random jammer: The jammer randomly jams a few frequency channels based on a specific probability distribution.

The jammer may choose the same or different jamming pattern to jam the transmissions at each AP. One AP will train the DRL model from scratch, and then transfer the learned model to other APs to help accelerate their DRL training process. The wireless network environment used for this AP to train its DRL model is called the source domain, whereas the network environment used for the other APs to train their DRL model is called target domain.

Chapter 4

The Proposed Anti-Jamming Method

The proposed anti-jamming methodology can be divided into three components, as depicted in Fig. 4.1. The first part is the DRL model which is the main algorithm used to learn an anti-jamming strategy in source and target domains. The second part is IFE-based TL which accelerates the learning process in target regions. The third part, named *feature relevance explanation* (FRE), introduces an idea to quantitatively measure the difference between a source and target domain, and based on that we can determine appropriate TL model settings.

4.1 Deep-Reinforcement Learning

The DRL model can be seen in Fig. 4.1. In a DRL model, the agent interacts with the environment by taking actions based on a policy and observing the results of its action. In

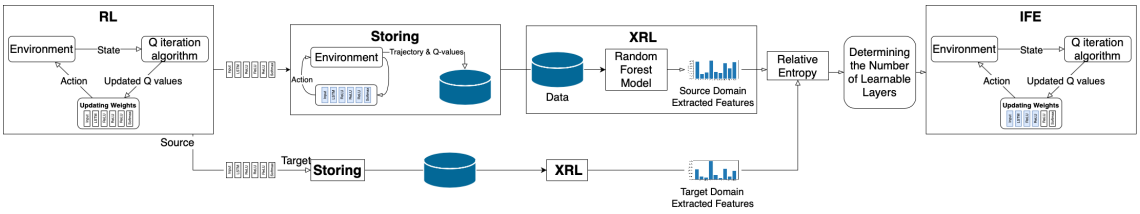


Figure 4.1: The Proposed Methodology

our DRL model, the policy is learned by a DNN.

We denote the observation after each signal transmission as O_i whose index i is a multiple of signal duration (τ time slots). We consider the observations to be the status of all channels during the signal transmission. We denote a channel's status at time slot t as $F_c(t)$, where c is the index of the channel, i.e., $c \in 1, 2, \cdot, N$. We can write observation in time slot $i = \lceil \frac{t}{\tau} \rceil$ as,

$$O_i = \begin{bmatrix} F_1(i-(\tau-1)) & F_1(i-(\tau-2)) & \cdots & F_1(i) \\ F_2(i-(\tau-1)) & F_2(i-(\tau-2)) & \cdots & F_2(i) \\ \vdots & & \ddots & \vdots \\ F_N(i-(\tau-1)) & F_N(i-(\tau-2)) & \cdots & F_N(i) \end{bmatrix}.$$

$F_c(t)$ in our system model can be one of the following four cases: 1) idle, 2) occupied by the user with successful transmission, 3) jammed with no user signal, and 4) disrupted signal (i.e., the user transmission is corrupted by the jammer). After observing the environment, the agent will generate states based on them as the input to a DNN. The DNN then tries to learn a policy following the *double Q-learning* algorithm [35]. Afterwards, the agent takes *actions* based on ϵ-*greedy* exploration [36]. The action $a \in \{C_1, C_2, \cdots, C_N\}$ is defined as the frequency channel that the active user in the AP's coverage region should use for transmission. After a user takes an action the agent will have a new observation which makes the *next state*. If the obtained next state is desired, i.e., the user has sent its packet successfully, the agent will be rewarded by 1. If a collision has happened because of that action, i.e., the user transmission is corrupted by the jammer, the agent will be punished by 0.1; therefore, we define the reward at time slot t as $r(t) = \begin{cases} 1 & \text{if successfull} \\ -0.1 & \text{otherwise} \end{cases}$. To formulate the state, we propose using the last K observations which can contain more jammer behavior information than only the last one

observation. Therefore, the state is defined as,

$$s(t) = [O_{t-(K-1)}, O_{t-(K-2)}, \cdots, O_t]. \tag{4.1}$$

K can be flexibly chosen based on the memory space of the AP. Subsequently, the double Q-iteration update is as follows:

$$Q_1(s(t), a(t)) \longleftarrow r(t) + \gamma Q_2(\underset{a' \in \mathcal{A}}{\text{argmax}} Q_1(s(t+1), a'(t+1))),$$

where γ is a discount factor, and $\mathcal{A}$ is the set of all possible actions; $\mathcal{A} = \{C_1, C_2, \cdots, C_N\}$. After l update iterations, Q_2 will be updated by $Q_2 \longleftarrow Q_1$. Lastly, the DNN can be trained with $s(t)$ as input and updated $Q_1(s(t), a(t))$ as the label.

We propose using RNNs as the neural network architecture since in our specific model, the input is time series. We use the LSTM layer [37] due to its best performance in handling time series input [4]. Moreover, we utilize dropout in our DNN architecture to reduce the calculations by randomly dropping out a specific percentage of the output of some layers in each iteration to speed up the training process. Dropout also helps avoid overfitting in neural networks. The proposed DNN architecture is as follows: LSTM with 8 units (neurons) and 10% dropout rate, ReLU [4] with 64 units and 10% dropout rate, ReLU with 32 and 10% dropout rate, ReLU with 16 units, and softmax with size equal to the number of actions.

4.2 IFE-based Transfer Learning

Although DRL is helpful, its learning process consumes considerable amount of time and energy. To accelerate the learning process with less time and energy, we deploy TL. The benefits of using TL are twofold: 1) instead of letting each AP learn the optimal DRL policy from scratch, one AP (in the source domain) can transfer the knowledge of the learned jammer behaviour to the other connected APs in the vicinity (i.e., target domain)

to enable them to employ the learned knowledge to speed up their learning process. 2) When the jammer changes its jamming behaviour, we can use TL to enable the DRL agent to quickly learn a new policy to adapt to such a change rather than taking a long time to learn a new policy from scratch.

TL techniques vary depending on the applications, and some TL methods, such as WI, do not differ much from a DRL model in terms of complexity. As the proposed method, we employ IFE to exploit its advantage in training a lower number of parameters in the target domain. So, when the target AP receives the transferred knowledge from the source domain in the form of a DNN parameters, before employing it, it will apply some changes by freezing some layers, using them as feature extractors, and training the other layers to learn the jammer pattern in case the jammer is using a different pattern in the target region. To do so, the AP in the target domain will decide how many layers of the transferred DNN should be frozen and how many learnable layers are needed. This decision depends on how much the jammer behaviour in the target domain differs from that in the source domain. As a result, in the target domain, we need information about the jammer behaviour in both the source and target environment.

Since the APs in the network do not communicate with the jammer, technically, the jammer is not an authorized part of the network, and the other parts do not have access to the jammer pattern. As a result, the only way to understand the jammer pattern is to estimate its actions by observing its jamming behaviour in the network. Apparently, an AP can only observe the jammer behaviour in its own coverage area. Transferring all the observed frequency spectra by the source AP to the target AP is impractical since such data can be significantly large. Furthermore, the transferred knowledge in TL, in the form of a DNN, is not easily explainable in the target AP. The reason is that a DNN is nothing more than a massive matrix in which its elements are considered as weights. So the policy stored in a DNN, usually in the form of the weights and the connectivity relation between units (neurons) of the DNN, is not understandable by the target AP to extract the piece of information related to the jammer behaviour. Therefore, we propose an explainable

artificial intelligence (XAI) [38] approach to interpret the DNN. From that, we can extract information about the jammer behaviour stored in the black box of the transferred DNN. The details of the proposed XAI method will be described in the subsequent section.

Using the proposed XAI method, we approximate how different the jammer behaviour is in the target domain from the one in the source domain. Using this information, we can decide the number of learnable layers. By constructing a DNN whose first few layers are from the transferred DNN learned in the source domain, and the last few layers are new layers with randomly initialized weights, the target AP can start the learning process of DRL in which the adopted layers from the transferred DNN are not being updated. Hence, the total number of learnable parameters can be significantly lower than in a regular DRL model, which leads to dramatically faster convergence.

The reason to adopt the first few layers from the transferred DNN but not the last few layers is that the first few layers store general feature information about the input data, and the last layers are more critical for a specified task [34].

Compared to the case where all the APs in different regions need to train a separate DRL model from scratch, the proposed TL method only requires one AP to train a complete DRL model from scratch, whereas all the other APs can use the transferred knowledge to train the DRL model with much fewer parameters and faster convergence, thus saving a significant amount of time and energy.

Depending on the differences between the source and target domains, the need for learning differs. We propose IFE to achieve flexible learning based on different needs. When the jammer behaviours in the target domain and in the source domain are slightly different, fewer learnable layers are needed. On the contrary, when the target domain is very different from the source domain, more learning would be required. In this case, fewer layers should be frozen and more learnable layers are needed because the more layers are frozen, the more skeptical the TL model will be.

4.3 Feature Relevance Explanation

Another challenge in this study is measuring how different the jammer behaves in the target domain from that in the source domain. One important fact about this measurement is that we do not have access to the jammer decision-making system. Hence, the only possible way is to estimate the difference. As described in the previous section, the information of jammer behaviour is not stored in the DNN explicitly. On the other hand, XAI techniques can help interpret the DNN decision-making process; from that, we can extract information about the jammer behaviour already learned by the DNN.

To our best knowledge, the first paper that has investigated the necessity of interpretability for AI techniques is [39], motivated by the European Union's new General Data Protection Regulation (GDPR) in the use of automated decision making algorithms in data protection law. [40] further investigated interpretability in machine learning. [38] has surveyed the challenges and applications of XAI in different machine-learning techniques. More specifically, [41] surveys the use of XAI in reinforcement learning. Among the different techniques listed in this survey, Linear Model U-Trees [42] are more useful in our system model, especially the feature extraction part.

The technique we use here in XAI taxonomy is called FRE [38], in which we try to extract the importance of the input space features. The reason is that the input space in our DRL model is the observed status of frequency channels. By extracting the importance of the features, we can figure out which channels are of higher importance during the DNN decision-making process. We propose using a *random forest* model [43] for extracting the features. The random forest model consists of several tree models such as *decision tree* (DT) models [43]. A sample of random forest DTs is illustrated in Fig. 4.2. In this figure, in each node, a condition/feature is considered (second line in the node). Gini index of each feature is on the third line and the fourth line represents the probability distribution of the next action. To interpret this specific DT, we start from the first node. The conditions/feature considered in this node by the random forest model is if channel 4

is disrupted or successfully occupied by the user. If either of these conditions holds, i.e., if channel 4 is occupied by the user successfully or disrupted by the jammer, it indicates the DRL agent's decision is to transmit through channel 4 with 100% certainty (node 17). If none of the conditions hold, i.e., channel 4 is idle or occupied by only the jammer, it will go to node 2. This will continue until the last node in the DT model. In a random forest model, m number of features among all features set, M, are randomly sampled. The feature resulting in the lowest *Gini index* will be chosen as the splitting criterion [44]. This process will be repeated until a DT is created. Afterward, other trees will be created by different feature samples. After getting all the trees, we calculate the importance of the features using the random forest trees.

We define $\mathcal{M}$ as the set of all tree nodes, $\mathcal{M}_i$ as the set of nodes splitting on feature i (F_i) and $\mathcal{C}_d$ as the set of child nodes of node d. Based on [45], the importance of the features in DT models can be calculated as,

$$I_{F_i} = \frac{\sum_{d \in \mathcal{M}_i} (w_d G_d - \sum_{z \in \mathcal{C}_d} w_z G_z)}{\sum_{d \in \mathcal{M}} (w_d G_d - \sum_{z \in \mathcal{C}_d} w_z G_z)}, \tag{4.2}$$

where $i \in \{1, 2, ..., N\}$, I_{F_i} denotes the importance of feature i, w_k and G_k denote the weighted number of samples reaching node k and the Gini impurity value of node k, respectively. From Equation (4.2), it can be implied that the purer children of a node are, meaning that the child nodes have low Gini impurity values, the more important the feature will end up with. The normalized importance of the features is given by $\mathcal{I}_{F_i} = \frac{I_{F_i}}{\sum_{j=1}^{L} I_{F_j}}$. Finally, the normalized importance of the features for a random forest model equals $\frac{\sum_{\mathcal{T}} \mathcal{I}_{F_i}}{|\mathcal{T}|}$, where $\mathcal{T}$ is the set of all trees in the random forest model and $|\mathcal{T}|$ is the cardinal number of set $\mathcal{T}$.

The proposed FRE method aims to quantitatively measure the difference between source and target domain environments. To this end, the source AP first extracts the importance of the environment features using a random forest model and transfers that information along with the DNN. Then, the target AP tests the transferred DNN in the

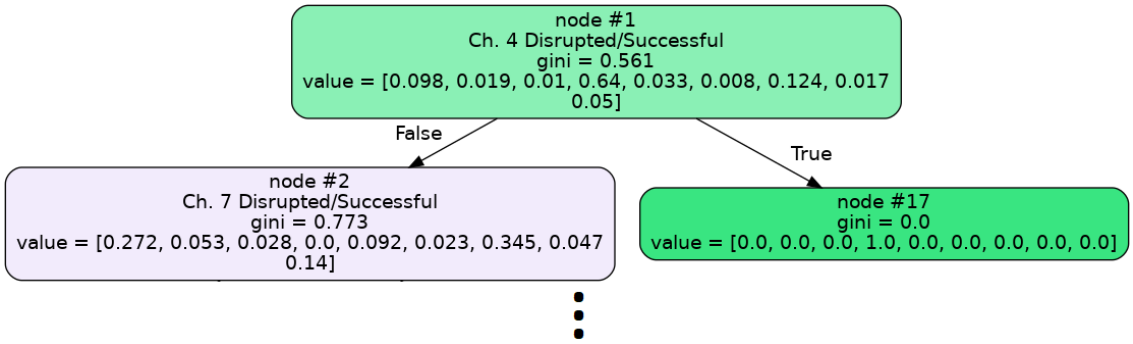


Figure 4.2: Part of a sampled DT from the random forest model.

new environment to gather enough data in the target domain and pass them to the random forest model.

Using the obtained data in the target domain, the target AP extracts the importance of the features out of the transferred DNN and compares them with those obtained in the source domain using relative entropy. Relative entropy can tell how distant the two importance of the features distributions are from each other, which can be used as a quantitative metric to measure the difference between the source and target domains. Relative entropy, also known as *Kullback–Leibler divergence*, of two probability distributions P and Q on the sample space $\mathcal{X}$ is

$$D_{\mathrm{KL}} = \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)}. \tag{4.3}$$

The reason for testing the transferred DNN obtained from the source domain in the target domain is that the importance of the features gives information about the DRL agent actions, not the jammer behaviour. So if a feature has different importances in source and target domains following the same policy (the same DNN), it means the policy learned from the source domain is not effective enough in the new environment (due to the change of jammer behaviour) and needs to be retrained.

Chapter 5

Numerical Results

To evaluate the proposed method, we simulate[1] the wireless network environment as follows. We consider 9 channels available at each AP. Each time slot lasts 1ms, and each active user signal transmission lasts 10ms. At any time, there is always an active user trying to send data to an AP. The other simulation parameters are listed in Table 5.1.

For the jammer behaviours, we consider four scenarios in our simulations, one as the source domain and the rest as target domains:

- *Source:* We consider a sweep type of jammer for this scenario which jams each channel for 7ms, and after each 2ms it starts jamming the next channel. In this scenario, the probability of channels being jammed is the same for all channels, hence it is uniformly distributed. This scenario is considered as the source domain, and the DRL model is trained from scratch under this scenario.

- *Case 1:* This is the most similar scenario to the source domain with a slightly different jamming pattern. The jamming duration for each channel is 9ms, and there is a delay of 6ms after sweeping all the frequency channels and before starting the next sweeping cycle. It is notable that the jamming probability in Case 1 is even between channels.

[1] This thesis's simulation Python source code is available online at https://github.com/SiavashBarqiJaniar/TransferLearningWirelessNetwork

Table 5.1: Simulation hyperparameters

Symbol	Definition	Assigned Value
N	The number of transmission channels/features	9
$F_i(t)$	Channel status/input feature for channel i at time slot t	idle/successful jammed/disrupted
K	Number of observations in a state	10
γ	Discount factor	0.9
ϵ	Exploration rate in DRL model	90%
	Exploration rate in TL models	50%
ϵ decay	ϵ discount factor	0.9994
l	Secondary DNN Q update rate in double Q-iteration algorithm	100 iterations

- *Case 2:* The jammer in this scenario follows a totally different pattern from the previous two cases. Considering the nine channels, the jammer jams each channel with [3, 18, 18, 3, 18, 17, 3, 17, 3]% probability distribution. The jammer jams 6 channels for 100ms, then reselects channels to jam following the probability distribution.

- *Case 3:* The reason for introducing this scenario is to illustrate the flexibility of our proposed methodology in adapting the appropriate number of frozen layers in the IFE-based TL algorithm. This case is similar to case 2. The difference is that in this case, the jammer jams each channel with the probability distribution [14, 2, 14, 11, 14, 14, 14, 14, 3]%.

- *Case 4:* The probability distribution designed for this case is [12.4, 12, 12, 12, 12, 13, 11.8, 7.2, 7.6]%.

Case 1 is the most similar one to the source case since it has identical jamming probability distribution (both uniform) and the same pattern, followed by Cases 2 and 3, while Case 4 is the most different from the Source case. The probability distributions for Cases 2 to 4 do not clearly show how different they will make the target environment from Source. It is

worth noting that changing the jammer probability distribution does not necessarily mean that the learned policy in a different environment will not work. The relation between a learned policy and the jammer probability distribution is difficult to be unveiled. Hence, finding jamming probability distributions to distinguish the difference between the target domain and the source domain, i.e., making the policy learned in the source domain have different level of effectiveness in the target domain, is not easy. However, without loss of generality and for the sake of simulation, to find jamming probability distributions making three different cases different from the source domain, we first extract the importance of the features in the source domain and then design the scenarios in a way with different KL divergences between the feature importance distribution and the jammer probability distribution. Among Cases 2 to 4, the KL divergence value in Case 2 is smallest whereas that in Case 4 is the largest, indicating that Case 4 is the most different from the source domain, followed by Case 3, and then Case 2. The purpose of designing such different cases is to show the capability of the proposed method in adapting to different environments with different levels of learning. It is worth noting that the DRL agents at the APs do not have the prior knowledge of the jammer pattern and the jammer probability distribution in all the cases.

We first evaluate the RNN structure and compare it with the CNN structure that is commonly adopted in many existing works [2, 3, 33] as well as another benchmark method, named *Random*, in which the AP randomly chooses a channel for the user to transmit. Fig. 5.1 shows the comparison result in the Source scenario. In Fig. 5.1, the y-axis represents the throughput of the wireless network, and the x-axis represents time. Note that the DRL models with RNN and CNN consume different time in each iteration due to their different computational complexities. To have a fair comparison, we have to illustrate the simulation time of each model in proportion to their time complexity. To do so, we first show the computational complexity in terms of the number of parameters for the RNN and CNN models in Table 5.2. As can be seen from Table 5.2, even though in CNN the smallest possible convolutional layer is used, it still has a higher number of parameters than the RNN

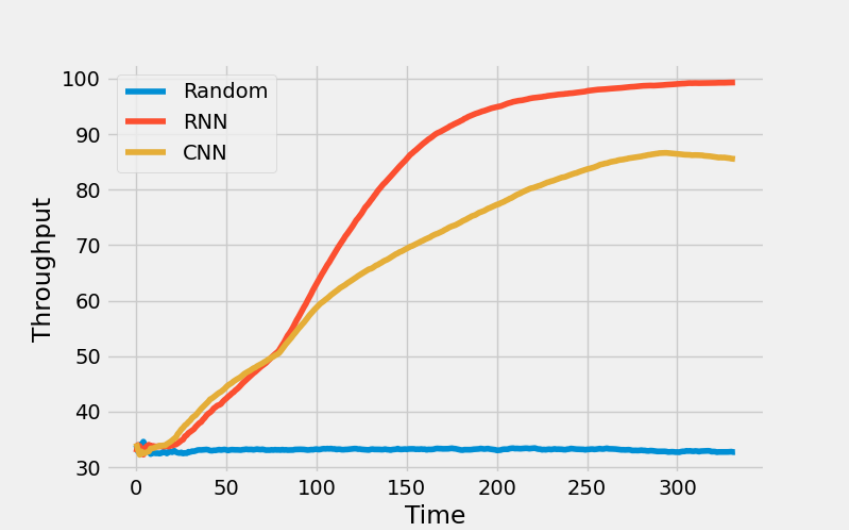


Figure 5.1: The performance of different DRL models and Random. Since Random does not have any learning parameters, it has not scaled in this figure.

model with the LSTM layer, while the utilized LSTM layer is not the smallest possible one[2]. From Table 5.2, we can see that the RNN has 777 parameters less than the CNN. Since the Random scheme has no parameters to be trained, we do not represent its result in the table. Then we use the number of iterations as the scale of the x-axis of Fig. 5.1 for the CNN model (i.e., the model with higher number of parameters) as a yardstick and shrink the scale for the RNN model in proportion to the ratio of its number of parameters to that in the CNN. Fig. 5.1 shows that the DRL model with RNN outperforms that with CNN structure, in terms of the convergence rate and the achieved throughput after convergence. Also, there is a big gap between the throughput of the Random scheme and DRL models after convergence. This justifies the advantage of DRL models even though they are more complex than the Random scheme.

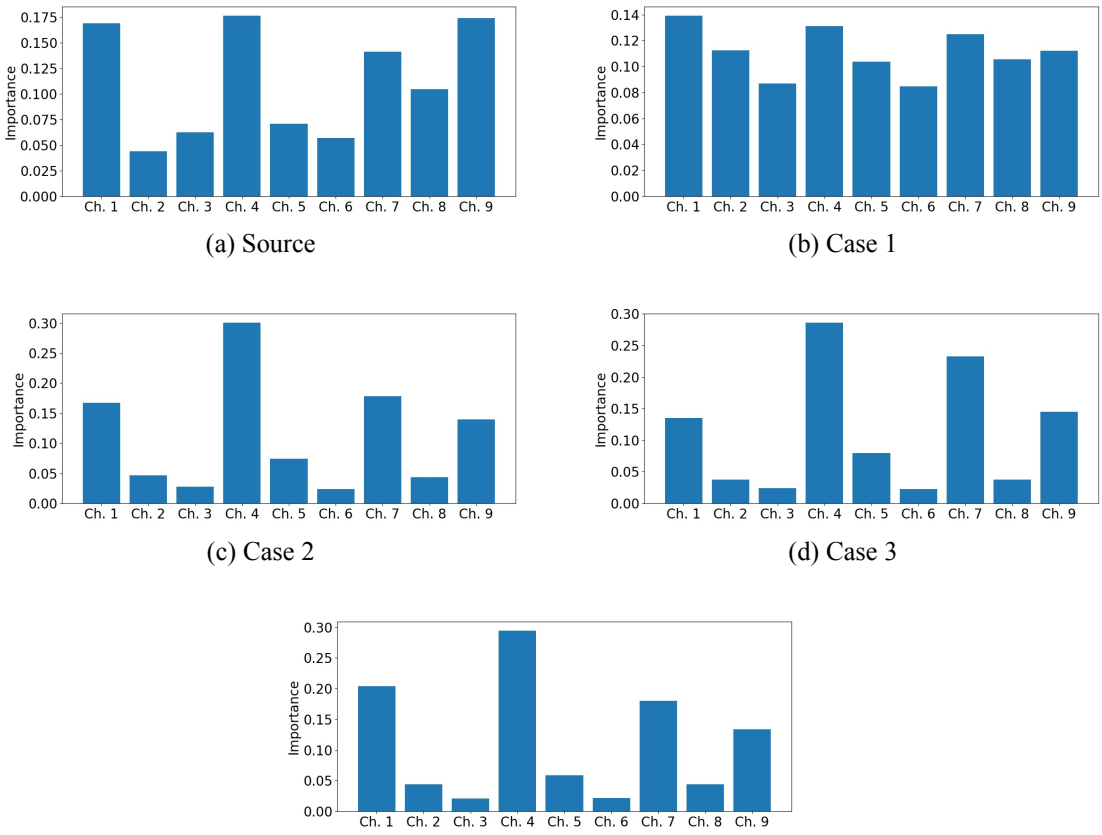
After the DRL model is trained at the first AP in the source domain, we extract the importance of the features of the source domain DNN in the source and target domains based on the FRE method, as described in Chapter 4. Fig. 5.2 shows the extracted importance

[2]We have not investigated the optimal LSTM size or DNN architecture for the considered system model as it is not the main concentration of this study.

Table 5.2: Training parameters of different models

Model	Layer	Number of parameters
RNN	LSTM	576
	ReLU	576
	ReLU	2080
	ReLU	528
	Softmax	153
	Total	3913
CNN	1D convolutional layer	1801
	ReLU	128
	ReLU	2080
	ReLU	528
	Softmax	153
	Total	4690

of the features in different scenarios, using the DNN trained in the source domain. Each bar indicates the importance of the channels from a scale of 0 to 1. Fig. 5.2a indicates that channels 1, 4, 7, and 9 are the most important, even though the jammer does not have any preference between different channels. We can model the sweep jammer probability distribution as uniform. The policy learned by the black box of AI (DRL model in our study) is hard to be interpreted as how such a policy is selected by the AI. As a result, it is hard to explain why channels 1, 4, 7, and 9 are more important from the AI's point of view, however, this policy leads to a significantly high throughput and performance. As shown in Fig. 5.2b, channels 1, 4, 7, and 9 still have the highest importance, but since the jammer in this scenario is less aggressive, the agent still has opportunities to select the rest of the channels. As a result, we can see from this figure that the other channels could achieve an importance value close to the aforementioned channels. In the rest of the figures, we can see that channels 1, 4, 7, and 9 have higher importance than the others, similar to Fig. 5.2a. It is because the tested DNN is the one that is trained in Source scenario. Even though it is the same DNN in all scenarios, we still see differences in the importance of features in different target domains. This will allow us to quantitatively estimate how much different



(a) Source

(b) Case 1

(c) Case 2

(d) Case 3

(e) Case 4

Figure 5.2: The importance of the features.

they are from the source domain.

We estimate the differences between the source and target domains, gathered in Table 5.3, by calculating the relative entropies based on the random forest results. As can be seen in Table 5.3, the estimation matches the expectation that Case 1 has the lowest relative entropy, followed by Case 2 and Case 3, and Case 4 has the highest value. The higher the relative entropy, the more different the target and source domains are.

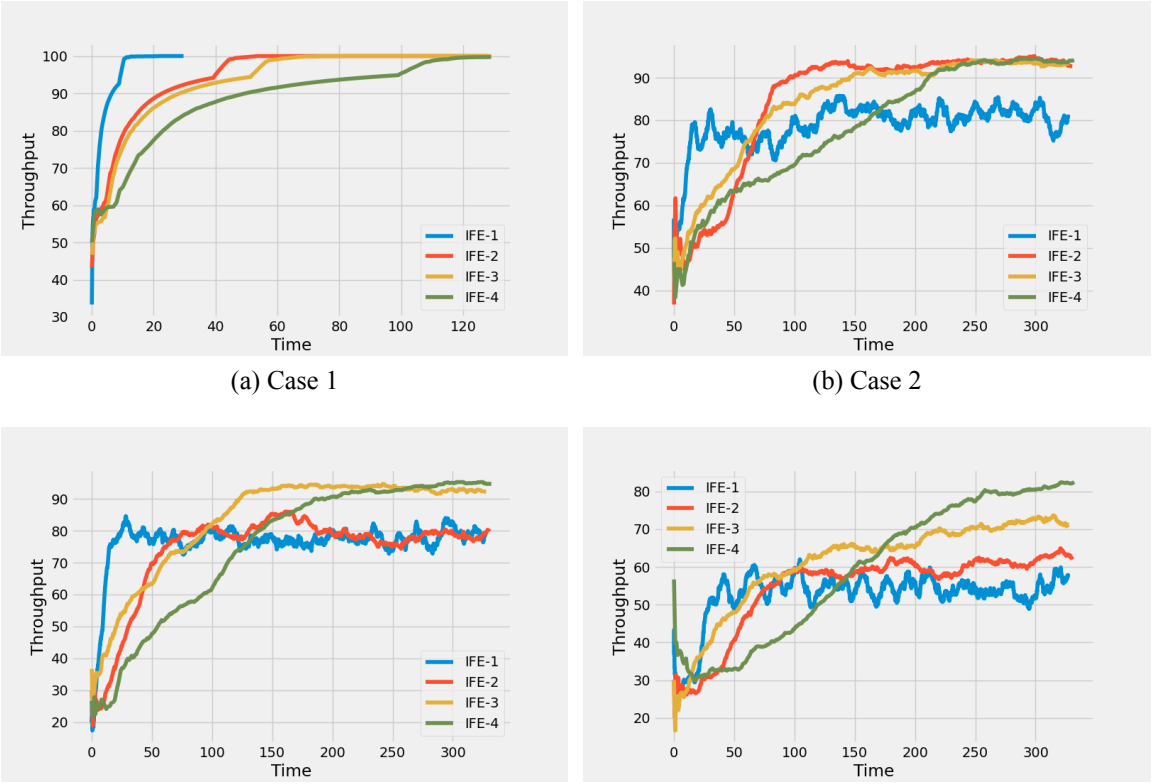
Defining a general algorithm determining the number of frozen layers from the estimated relative entropy values is challenging. Nevertheless, to see the optimum number of layers to be frozen, we test the performance of different possible layer selections to freeze and compare the throughput and convergence rate, as shown in Fig. 5.3. Similar to Fig.

Table 5.3: Relative entropy values obtained from the simulation.

Target Domain	Case 1	Case 2	Case 3	Case 4
relative entropy values	0.1231	0.1619	0.1833	0.1945

5.1, we scale the x-axis for each curve in proportion to their computational complexity in each iteration. IFE-X in the graph means the last X layers of the RNN are learnable and the rest are frozen. As can be seen, for Case 1 (Fig. 5.3a), IFE-1 converges faster than the other IFE models since it is the most similar scenario to Source and the jammer follows the same pattern. As a result, the optimal policy in Case 1 and Source are close and not much learning is needed in the target domain (Case 1). For Case 2 (Fig. 5.3b), it is IFE-2 that works the best. Even though Case 2 has close probability distribution to the Source importance of features distribution, since it has a totally different jamming pattern, it needs more layers to be trainable. Considering the similarity to the source domain, Case 3 is a scenario in the middle (Case 1 is the closest one to the source domain, followed by Cases 2, 3, and 4). In this case, IFE-3 outperforms the others, while in Case 4, IFE-4 is the supreme model meaning that more learnable layers are needed. In Fig. 5.3c, both IFE-3 and IFE-4 converge to higher throughput, outperforming IFE-1 and IFE-2. However, between IFE-3 and IFE-4, IFE-3 obviously surpasses in terms of the convergence rate because of its lower complexity. Fig. 5.3d indicates that even though freezing more layers is time and energy efficient, when the target model is too different from the source domain, models with fewer learnable layers do not have enough parameters to find the optimum policy. As a result, we can see that IFE-1, IFE-2, and IFE-3 converge to a value significantly lower than IFE-4. It is worth noting that Figs. 5.3b, 5.3c, and 5.3d do not converge to 100% throughput in contrary to Case 1 since the jammer takes more random decisions in Cases 2 to 4. Random decisions are not predictable by ML models as they do not follow any certain pattern, as a result, there is nothing except estimating the probability of the random decisions to be learned by the DRL model. By comparing the results in

Fig. 5.3 and Table 5.3, we can see that the relative entropy value is an important indicator that can guild us to appropriately choose the number of learnable layers in the IFE method, i.e., the higher the relative entropy value, the more learnable layers should be chosen.



(a) Case 1

(b) Case 2

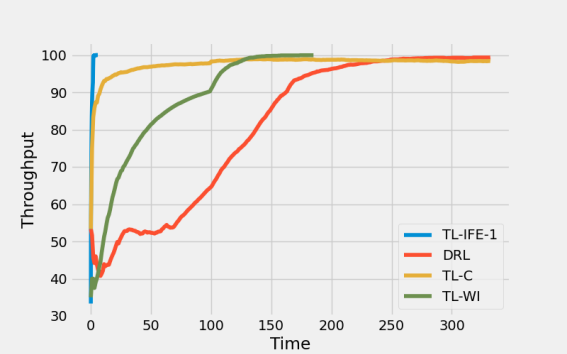
(c) Case 3

(d) Case 4

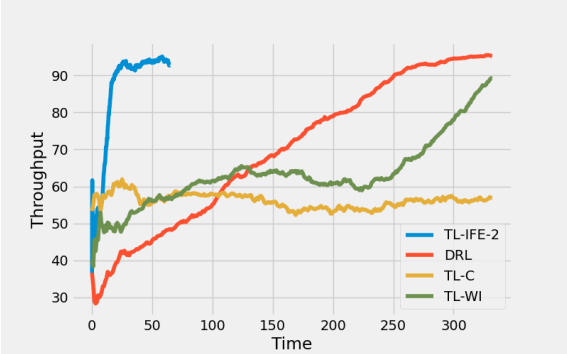
Figure 5.3: IFE based TL models convergence rates for different scenarios

We further compare our proposed TL method with other existing TL methods as well as the benchmark method, i.e., training the DRL model from scratch (marked as DRL in Fig. 5.4) under the four considered cases, as illustrated in Fig. 5.4. We consider *classifier* TL (marked as TL-C in the figure) [46], weight initialization TL (marked as TL-WI in the figure and explained in Chapter 2). In TL-C, decisions are naively made based the transfered DRL without any training. For each case, we choose the IFE model that offers the best performance as our proposed TL method. From Fig. 5.4, we can see that all the TL methods converge faster than the traditional method, i.e., training the DRL model

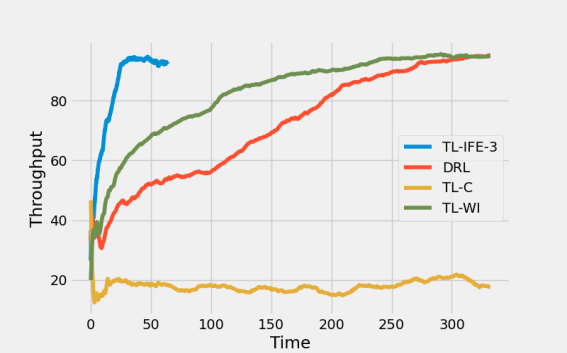
from scratch. However, some TL methods may not achieve good throughput after convergence. It is apparent that TL-C is the fastest and simplest model among others since there is no learning occurring in TL-C. In Fig. 5.4a, we can see that TL-C performs well in Case 1 since Case 1 is similar to the source domain, yet, it is still not able to achieve the throughput as high as what the other models have achieved after convergence since it does not learn anything from the new environment. Also, in Figs. 5.4b, 5.4c, and 5.4d, it can be seen that TL-C achieves a throughput significantly lower than all the other models. It is clear from Fig. 5.4 that our proposed TL method (IFE-based TL) outperforms the other TL methods and the traditional method that trains the DRL model from scratch in all considered scenarios.



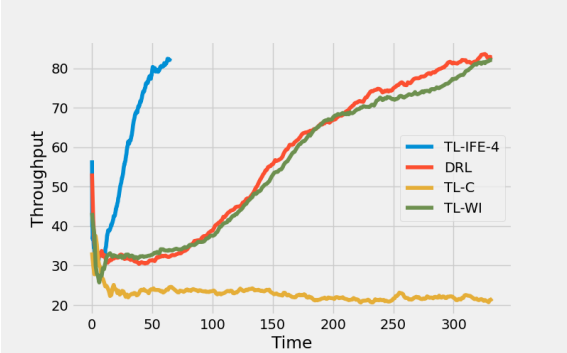
(a) Case 1



(b) Case 2



(c) Case 3



(d) Case 4

Figure 5.4: Different ML models convergence rates for different scenarios

Chapter 6

Conclusion

6.1 Summary of the Thesis

We have studied a time-efficient anti-jamming method in wireless networks where the jammer attempts to disrupt multiple communication regions. We have discussed the advantage of using RNNs in wireless network system models and compared it with a DRL model with CNN. However, training a DRL model from scratch in each regional AP can be considerably long. To accelerate the training process of the DRL model, we have proposed utilizing IFE-based TL due to its flexibility in adapting to the target environment depending on how different the jammer behaves in the target domain compared to the source environment. To measure the difference between the source and target domains, we have proposed an FRE-based technique using a random forest model. Experiment results show that our proposed TL method can effectively speed up the training process of the DRL model and outperforms the other TL methods.

6.2 Future Directions

In future directions, one can study a jamming scenario where more than one active user attempt to communicate through the available frequency channels. This problem is a *re-*

source allocation/management which is an existing hot research area in wireless networks and many researches are being done to do the resource allocation in its fairest and optimum way using ML and RL. Also, the exact relation between relative entropy values and the optimal number of learnable layers is yet to be found. It is worth noting that relative entropy bounds are $[0, \infty)$ while the number of learnable layers in a DNN is finite and dependant on the DNN architecture. The proposed method can only measure the difference between the source and target domains, and use that as a guideline for appropriate tuning settings in TL. However, we cannot guarantee that the selected tuning setting is optimal. Finding the precise relationship between the measured data and the corresponding optimal tuning configurations is challenging and deserves further study. In this thesis, we only utilized one of many kind of XAI techniques. One open challenge is to study different types of XAI which provide different ways to explain the behaviour of the DRL model in different source and target domains.

Bibliography

[1] M. Wilhelm, I. Martinovic, J. B. Schmitt, and V. Lenders, "Short paper: Reactive jamming in wireless networks: How realistic is the threat?" in *Proceedings of the fourth ACM conference on Wireless network security*, 2011, pp. 47–52.

[2] X. Liu, Y. Xu, L. Jia, Q. Wu, and A. Anpalagan, "Anti-jamming communications using spectrum waterfall: A deep reinforcement learning approach," *IEEE Communications Letters*, vol. 22, no. 5, pp. 998–1001, 2018.

[3] J. Xu, H. Lou, W. Zhang, and G. Sang, "An intelligent anti-jamming scheme for cognitive radio based on deep reinforcement learning," *IEEE Access*, vol. 8, pp. 202 563–202 572, 2020.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[5] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[6] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. USA: Wiley-Interscience, 2006.

[7] Radio jamming. [Online]. Available: https://en.wikipedia.org/wiki/Radio_jamming#cite_note-3

[8] K. Grover, A. Lim, and Q. Yang, "Jamming and anti-jamming techniques in wireless networks: A survey," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 17, no. 4, pp. 197–215, 2014.

[9] S. Khattab, D. Mosse, and R. Melhem, "Jamming mitigation in multi-radio wireless networks: Reactive or proactive?" in *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, 2008, pp. 1–10.

[10] A. D. Wood, J. A. Stankovic, and G. Zhou, "Deejam: Defeating energy-efficient jamming in ieee 802.15. 4-based wireless networks," in *2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*. IEEE, 2007, pp. 60–69.

[11] V. Navda, A. Bohra, S. Ganguly, and D. Rubenstein, "Using channel hopping to increase 802.11 resilience to jamming attacks," in *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*. IEEE, 2007, pp. 2526–2530.

[12] R. Gummadi, D. Wetherall, B. Greenstein, and S. Seshan, "Understanding and mitigating the impact of rf interference on 802.11 networks," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 385–396, 2007.

[13] B. Kerkez, T. Watteyne, M. Magliocco, S. Glaser, and K. Pister, "Feasibility analysis of controller design for adaptive channel hopping," in *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, 2009, pp. 1–6.

[14] H. Wang, L. Zhang, T. Li, and J. Tugnait, "Spectrally efficient jamming mitigation based on code-controlled frequency hopping," *IEEE Transactions on Wireless Communications*, vol. 10, no. 3, pp. 728–732, 2011.

[15] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[16] R. Zhao, J. Xia, Z. Zhao, S. Lai, L. Fan, and D. Li, "Green mec networks design under uav attack: A deep reinforcement learning approach," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 3, pp. 1248–1258, 2021.

[17] Y. S. Nasir and D. Guo, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2239–2250, 2019.

[18] H. A. Al-Rawi, M. A. Ng, and K.-L. A. Yau, "Application of reinforcement learning to routing in distributed wireless networks: a review," *Artificial Intelligence Review*, vol. 43, pp. 381–416, 2015.

[19] C. Zhong, M. C. Gursoy, and S. Velipasalar, "Deep reinforcement learning-based edge caching in wireless networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 1, pp. 48–61, 2020.

[20] S. O. Somuyiwa, A. György, and D. Gündüz, "A reinforcement-learning approach to proactive caching in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1331–1344, 2018.

[21] L. Xiao, Y. Li, G. Han, G. Liu, and W. Zhuang, "Phy-layer spoofing detection with reinforcement learning in wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 10 037–10 047, 2016.

[22] Q. Liu, L. Cheng, A. L. Jia, and C. Liu, "Deep reinforcement learning for communication flow control in wireless mesh networks," *IEEE Network*, vol. 35, no. 2, pp. 112–119, 2021.

[23] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for distributed dynamic spectrum access," *IEEE transactions on wireless communications*, vol. 18, no. 1, pp. 310–323, 2018.

[24] L. Kang, J. Bo, L. Hongwei, and L. Siyuan, "Reinforcement learning based anti-jamming frequency hopping strategies design for cognitive radar," in *2018 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*. IEEE, 2018, pp. 1–5.

[25] L. Bommisetty and T. Venkatesh, "Resource allocation in time slotted channel hopping (tsch) networks based on phasic policy gradient reinforcement learning," *Internet of Things*, vol. 19, p. 100522, 2022.

[26] S. B. Janiar and V. Pourahmadi, "Deep-reinforcement learning for fair distributed dynamic spectrum access in wireless networks," in *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2021, pp. 1–4.

[27] S. Barqi Janiar and V. Pourahmadi, "Deep-reinforcement learning for fair distributed dynamic spectrum access in priority buffered heterogeneous wireless networks," *IET Communications*, vol. 15, no. 5, pp. 674–682, 2021.

[28] G. Han, L. Xiao, and H. V. Poor, "Two-dimensional anti-jamming communication based on deep reinforcement learning," in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2017, pp. 2087–2091.

[29] S. Ak and S. Brüggenwirth, "Avoiding jammers: A reinforcement learning approach," in *2020 IEEE international radar conference (RADAR)*. IEEE, 2020, pp. 321–326.

[30] A. Rossi, M. Hagenbuchner, F. Scarselli, and A. C. Tsoi, "A study on the effects of recursive convolutional layers in convolutional neural net-

works," *Neurocomputing*, vol. 460, pp. 59–70, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231221010559

[31] G. Chen, Y. Zhan, Y. Chen, L. Xiao, Y. Wang, and N. An, "Reinforcement learning based power control for in-body sensors in wbans against jamming," *IEEE Access*, vol. 6, pp. 37 403–37 412, 2018.

[32] C. Han and Y. Niu, "Multi-regional anti-jamming communication scheme based on transfer learning and q learning," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 13, no. 7, pp. 3333–3350, 2019.

[33] X. Lu, L. Xiao, C. Dai, and H. Dai, "Uav-aided cellular communications with deep reinforcement learning against jamming," *IEEE Wireless Communications*, vol. 27, no. 4, pp. 48–53, 2020.

[34] C. T. Nguyen, N. Van Huynh, N. H. Chu, Y. M. Saputra, D. T. Hoang, D. N. Nguyen, Q.-V. Pham, D. Niyato, E. Dutkiewicz, and W.-J. Hwang, "Transfer learning for future wireless networks: A comprehensive survey," *arXiv preprint arXiv:2102.07572*, 2021.

[35] H. Hasselt, "Double q-learning," *Advances in neural information processing systems*, vol. 23, 2010.

[36] S. Levine, "Cs 294: Deep reinforcement learning," 2018.

[37] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[38] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information fusion*, vol. 58, pp. 82–115, 2020.

[39] B. Goodman and S. Flaxman, "European union regulations on algorithmic decision-making and a "right to explanation"," *AI magazine*, vol. 38, no. 3, pp. 50–57, 2017.

[40] Z. C. Lipton, "The mythos of model interpretability. queue 16, 3, article 30 (june 2018), 27 pages," 2018.

[41] E. Puiutta and E. Veith, "Explainable reinforcement learning: A survey," in *International cross-domain conference for machine learning and knowledge extraction*. Springer, 2020, pp. 77–95.

[42] G. Liu, O. Schulte, W. Zhu, and Q. Li, "Toward interpretable deep reinforcement learning with linear model u-trees," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2019, pp. 414–429.

[43] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[44] Statquest: Random forests part 1 - building, using and evaluating. [Online]. Available: https://www.youtube.com/watch?v=J4Wdy0Wc_xQ

[45] S. Ronaghan. The mathematics of decision trees, random forest and feature importance in scikit-learn and spark. [Online]. Available: https://towardsdatascience.com/the-mathematics-of-decision-trees-random-forest-and-feature-importance-in-scikit-learn-and-spark-f2861df67e3

[46] Transfer learning in keras with computer vision models. [Online]. Available: https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/