Bridging the Gap Between Simulation and Reality on WMN Configuration via Deep Learning

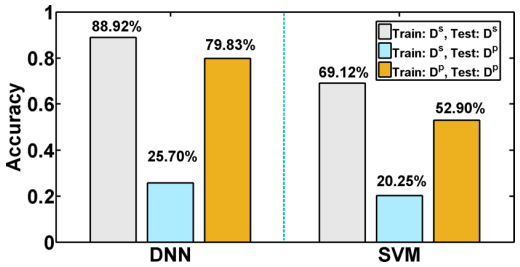
Xia Cheng xcheng@fiu.edu

1. Introduction

This report tends to basically introduce my project, which focuses on bridging the gap between simulation and reality on wireless mesh network configuration via deep learning method.

In the real-world industrial facilities, a great number of wireless mesh networks (WMNs) have been deployed for industrial automation. For example, over the last 15 years, more than 54 thousand WirelessHART networks have been deployed globally by Emerson Process Management to monitor and control industrial processes. Although WMNs work satisfactorily most of the time, they are often difficult to configure because configuring a wireless mesh network is a complex process, involving theoretical computation, simulation, and field testing, among other tasks. Simulating a wireless mesh network provides distinct advantages than experimenting on a physical network when it comes to identifying a good network configuration: a simulation can be set up in less time, introduce less overhead, and allow for different configurations to be tested under the same conditions. Therefore, many wireless simulators, such as TOSSIM, Cooja, and NS-3, have been developed and used for wireless mesh network configuration.

However, it is still very challenging to date to set up a simulation that captures extensive uncertainties, variations, and dynamics in real-world WMN deployments. According to the experiment results in our previous paper, shown in the following figure, there is a great gap between the simulation data and the physical testbed data.

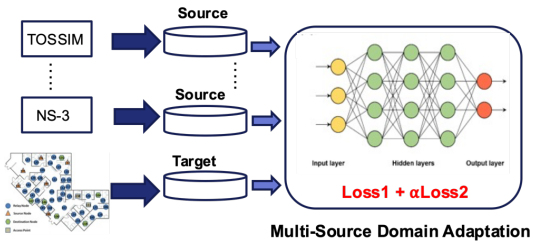


In this project, I plan to close the gap between simulation and reality via multi-source domain adaptation.

Multi-source domain adaptation aims to learn from multiple source domains, together with or without a target domain, and then generate a model that performs well on the target domain. The assumption is that the source and target domains are associated with the same label space. It has been successfully used in computer vision and natural language processing. The existing studies on those applications have shown that multi-source domain adaptation can mitigate the harmful effects of domain discrepancy.

2. Design and Implementation

The project goal is to use a large amount of simulation data generated by different simulators together with a small amount of physical data to train a classifier, which accurately predicts the configuration of a physical wireless network.



2.1 Design

As the above figure shows, the data traces from different network simulators and the physical testbed are collected separately. After processing the raw data into source domain data and target domain data, I get the data sets for the deep learning method. I choose a DNN model for my multi-source domain adaptation method, which consists of three layers. I employ 120 neurons in the first hidden layer and 84 neurons in the second hidden layer. I select 88 neurons in the output layer.

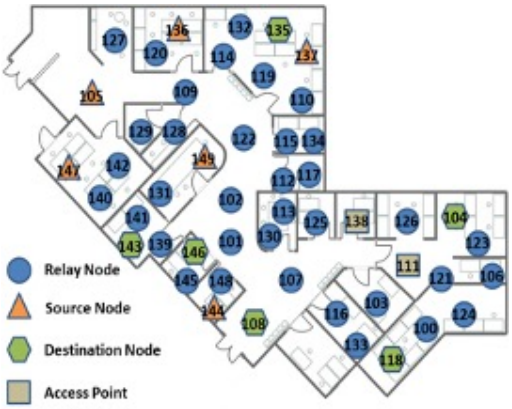
I employ two loss functions for the proposed multi-source domain adaptation method. The first loss function is used to learn the knowledge about the target domain from a few amount of target domain data traces. The second loss function is used to help the classifier learn from a larger amount of simulation data traces generated by different simulators. The coefficient α achieves the balance between the first loss on target domain data and the second loss on source domain data. In the end, the method generates the classification for the input data.

2.1 Data Collection

To collect data traces, I first run an open-accessible WirelessHART network implementation on our testbed and configure six different data flows as the following table shows.

Flow	Sensor	Actuator	Data Period	Priority
1	147	146	500ms	1
2	144	143	500ms	2
3	105	104	500ms	3
4	149	118	1,000ms	4
5	136	135	1,000ms	5
6	137	108	1,000ms	6

Then, I measure the network performance, including the end-to-end latency, the battery lifetime, and the reliability, under each network configuration. There are 88 distinct network configurations, I collect 75 data traces under each network configuration. Totally, I collect 6,600 data traces for the target domain.



The above figure plots the deployment of our testbed, which consists of 50 nodes. Each node is composed of a TelosB device, a Raspberry 3, and a battery box, as the following figure shows.

Then I adopt the same WirelessHART network implementation and the data flow settings on two simulators, TOSSIM

and Cooja. I simulate the network operation and measure the performance under each network configuration. I also collect 6,660 data traces from each simulator.

2.3 Implementation

I select python as the programming language for my project and choose Pytorch as the framework to run the multi-source domain adaptation model. The developing platform is Mac OS Catalina.

```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
from tqdm import tqdm
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import mmd
from pandas import Series,DataFrame
import matplotlib.pyplot as plt
import seaborn as sns
from variable import *

X = pd.read_csv("res1.csv", sep=',')
Y = pd.read_csv("res2.csv", sep=',')
print(X.describe())
#print(X['T4'])
print(Y.describe())
#print(Y['T4'])

data1 = X['T4'].values.tolist()
#print(data1)

data2 = Y['T4'].values.tolist()
#print(data2)
print(len(data1))
print(len(data2))
Same_rec = [x for x in data2 if x in data1]
Diff1_rec = [x for x in data2 if x not in data1]
Diff2_rec = [x for x in data1 if x not in data2]
print(len(Same_rec))
#print(Same_rec)
print(len(Diff1_rec))
print(len(Diff2_rec))
```

This above figure shows part of the implementation, which are used to analyze the distribution difference between different simulation data sets. It can calculate the number of data traces that are shared and not shared by these sources. Then the program can compute the coefficient used in the loss function accordingly.

3. Evaluation and Conclusion

I perform a series of experiments to evaluate the prediction accuracy achieved by the proposed method and compare it against the accuracy of the state-of-the-art baseline.

My method achieves 80.45% accuracy on the network configuration prediction when using 440 target data traces, 3,300 TOSSIM data traces, and 3,300 ns-3 data traces. As a comparison, the baseline provides 70.56% accuracy when using the same amount of training data. Moreover, the execution time consumed by my method is comparable to the running time of the baseline.

The experiments in the project show it is beneficial to use the simulation data generated by multiple simulators as the training data. Compared to the baseline, the multi-source domain adaptation method implemented in my project can enhance the prediction accuracy from 70.56% to 80.45%. Therefore, multi-source domain adaption can bridge the gap between simulation and reality on wireless network configuration.

4. Acknowledgements

For the whole semester, I would like to thank Professor Mo Sha, for his great help and some hints when I face any barriers.