Master Degree Thesis

Master's degree in Communications and Computer Networks Engineering

---

Assessment of Domain Adaptation Approaches for QoT Estimation in Optical Networks

---

Politecnico di Torino

Supervisor:
Andrea Bianco

Co-supervisors:
Cristina Rottondi
Alessandro Giusti

Author:
Riccardo di Marino
241826

December, 2019

A thesis submitted in fulfillment of the requirements for the
Master's degree in Communications and Computer Networks Engineering

written in

Telecommunication Networks Group
@ Politecnico di Torino,
Department of Electronics and Telecommunications
@ Politecnico di Torino

Abstract

Assessment of Domain Adaptation Approaches for QoT Estimation in Optical Networks

Predicting the Quality of Transmission (QoT) of a candidate lightpath prior to its establishment plays a pivotal role for an effective design and management of optical networks. In the last few years, supervised Machine Learning (ML) techniques have been advocated as promising approaches for QoT estimation, but to ensure the effectiveness of their training phase, a large amount of samples (training set) must be provided to the learning algorithm. Unfortunately, the collection of training samples is often hindered by practical issues (e.g., lack of dedicated telemetry equipment in every network node) or is too costly to permit the acquisition of large datasets. However, it is sometimes possible to rely on large training datasets from a different network (source domain) than the one on which the ML model operates (target domain). In such a scenario, we wish to exploit at best the data from the source domain to tailor a good model to the target domain. This approach is known in ML research as Domain Adaptation (DA). Note that most of the existing DA techniques require to complement the dataset from the source domain with a few samples from the target domain: quantifying the amount of samples extracted from the target domain needed to achieve satisfactory predictive performance of the adopted learning model is a crucial issue to determine the practical applicability of DA techniques in real-world scenarios. In this thesis, we evaluate the effectiveness of two existing DA approaches (i.e., feature augmentation and domain adaptation) for ML-based QoT estimation of candidate lightpaths, for a fixed/variable number of available training samples from the source/target domain. As source/target domains, we consider several network topologies with varying transmission equipment characteristics. Results show that, when the number of samples from the target domain is very limited (e.g., in the order of tens), DA approaches consistently outperform standard supervised ML techniques.

KEYWORDS **Machine learning**, **Transfer learning**, **Domain Adaptation**, **Quality of Transmission**, **Bit Error Rate**

Acknowledgements

For electronic submission of this elaborate the page has been left blank.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Optical network systems have emerged as the best physical infrastructure to accommodate large traffic volumes generated by new applications, e.g., cloud computing, but they require to ensure to the application layer services the quality of transmission (QoT) of the data flowing through the network. It could happen that a particular service request is affected by degradation due to the lack of resources needed to ensure correct transmission and has to be rerouted, or even blocked. For this reason, it is crucial to allow the optimal support for the services that will leverage the underlying network.

Predicting the QoT of a candidate lightpath, i.e a path between two nodes of the network in which the optical signal can pass through without electronic conversion, prior to its establishment plays a pivotal role for an effective design and management of optical networks.

The approaches that since years have tried to solve this kind of issue are either *mathematical models*, which approximate the behavior of the network under different conditions, or by the use of *simulation frameworks*, that instead are able to emulate the propagation of the optical signal along the fiber core; the former approach, very often, leads to a not well-designed problem and introduce high margins in the computations as a consequence of the initial uncertainties on input parameters and simplified assumptions made at the beginning; the second one requires high computational effort when applied to a realscale scenario. Both, furthermore, leads to over-provisioning in the network design, thus incurring in increased costs.

In the last few years, Machine Learning (ML) is finding a lot applications in optical networks, especially in cross-layer frameworks, i.e scenarios in which the physical layer can interacts and trigger changes at network layer by the analysis of transmission performance measurements such as the Bit Error Rate (BER) [13]; and its uses as technique for QoT estimation have been considered as promising alternatives to the traditional methods.

Supervised ML-based approaches are used to learn a mapping function which is able to infer the value of an output variable from a vector of input features, e.g., properties of the lightpath such as length, amount of served traffic and adopted modulation format. The output variable coincides with the expected QoT measure along the lightpath, and it could be identified, for example, by the Bit Error Rate (BER). The collection given as input to the ML algorithm is composed by samples which are the *features vector* of an established lightpath. Each of these samples is associated with the actual value of the *output variable*, measured at the receiver.

An effective learning phase can be achieved if a large training set, i.e. a collection of features vectors, is provided to the learning algorithm.

On one hand, a large training set provides a general view of the already established lightpaths and it is representative of the samples that will be used to train the model and exploit it to predict the QoT. On the other hand, it is not so trivial to have a large collection, especially because many times, the costs to acquire large datasets are very high or there are some issues due to the lack of the necessary telemetry equipment in the nodes of the network.

In a network in phase of deployment, it results difficult to make an efficient monitoring, especially because the number of installed lightpaths is limited; as a consequence, the collection of samples useful to build a wide training set is not possible. However, data could be collected following a different approach: i.e, relying on another network, for instance, in which a large number of lightpaths are already deployed and available to obtain a dataset from which the ML model can learn.

Under some conditions, we are able to have a large amount of data S associated to a *source domain*, i.e. monitoring a particular backbone network for a long period, and train a model for predicting our target value, i.e. the QoT of the lightpaths that have to be established into a different domain: the *target domain*. The target domain is a newly deployed network that has a limited availability of labeled data T. In this case, our purpose is to exploit the data extracted from the source domain to fit, in the best way, a model to the target domain.

This kind of approach in ML research is know as *Domain Adaptation* (DA) and it is based on the fact that samples belonging to a particular domain provide useful information to other domain. In this thesis we investigate on the effectiveness of this approach, when applied to the QoT estimation problem.

Intuitively, the accuracy of the prediction operated on the candidate lightpaths of the target domain is affected by various aspects related to the difference between the two domains (e.g., the type of fibers that are installed or the distribution of the length of the lightpath). The more the target domain differs from the source domain, the less accurate the prediction is expected to be.

Indeed, the mapping between input features and target variable changes if applied in the source or in the target domains. This is due to the fact that, formally, the joint probability distribution of features vector and output variable differs in the two domains.

Therefore, training samples coming from the target domain have to be somehow integrated with samples of the source domain; this step is crucial, because it is impossible to quantify first the amount of target data that ensure to achieve high and accurate predictive performance. For this reason it results difficult to determine the applicability of DA techniques in real-world scenarios.

To evaluate the effectiveness of existing DA approaches for ML-based QoT estimation of candidate lightpaths, we consider two different network topologies characterized by either links of the same fiber type and same transmission equipment, or by different fiber and amplifier types.

In order to conduct experiments in different scenarios and exploit more combinations of source and target domain, we redesign the two topologies, multiplying each of the links by a particular scaling factor. This way, we obtain 24 combinations of source and target domains to investigate about the different performance achieved by DA.

We assess the performance of two type of DA techniques comparing the performance in the different cases, where we change the number of samples belonging to the target domain. Besides, we investigate, on the degree of dissimilarity in between the two domains, quantified as *intersection metric* and relate it to the different performance achieved by DA, considering various combinations of source and target domains.

1.2 Thesis outline

The rest of the thesis is organized as follows: Chapter 2 presents a review of the related literature that exploits transfer learning and its application in optical networks for QoT estimation. It provides a general overview of the current research directions and highlights similarities and dissimilarities with respect to our work.

Chapter 3 reviews the key concepts to understand the theory on which the methods, approaches and models we use to perform experiments are based. We provide a ML basic overview with an explanation of the types of learning models and the description of the algorithms adopted in this work. Then, the concept of transfer learning it is introduced and in particular the DA schemes, focusing on implementation details. At the end of the chapter, the tool we use to synthetically generate the dataset belonging to a particular domain and evaluate the lightpath BER is also described.

In Chapter 4 the framework built to conduct our experiments is described in detail. We discuss the assumptions on the compositions of datasets S and T (i.e, the training

datasets collected from the source and target domains and the testbed built in order to perform DA, providing two variants.

In Chapter 5 we discuss and assess the results obtained from the numerical analysis, first giving a description of the datasets and then focusing on the evaluation of the performance metrics in the different considered scenarios.

Chapter 6 includes a conclusive section and summarizes the objectives achieved from the assessment of the results. Furthermore, it contains other considerations based on the developed experiments.

Chapter 2

Related Work

This chapter provides the description of recent studies that focus on the exploitation of transfer learning techniques in optical transmission systems in order to make predictions on various parameters that affect the quality of transmission. At the end of these descriptions, similarities and differences with respect to the work developed in this thesis will be highlighted.

2.1 Transfer Learning in literature

Recently, a few studies have appeared which propose the use of *transfer learning* to predict various optical network parameters. These studies highlighted the ability of transfer learning approaches to achieve good results in terms of performance, number of samples used to perform the training step and computational burden.

More in detail, some applications of TL are:

- QoT prediction in Real-Time Mixed Line-Rate Systems;
- OSNR estimation;
- Spectrum Optimization for the Resource Reservation in Space Division Multiplexing Elastic Optical Networks;

2.1.1 QoT prediction in Real-Time Mixed Line-Rate Systems

In [11], the prediction of quality of transmission (QoT) in a real-time mixed line-rate system is obtained by an artificial neural network (ANN) based transfer learning framework. The QoT prediction in optical transmission systems can be managed well by supervised learning algorithms because they are good to discover relations among input data and output data. The goal of the authors is to train a regression model to predict the Q-factor from the different domains, time by time, without making a classification based on QoT ranges.

The machine learning model is the artificial neural network that is a mathematical model of the human brain. It consists of neurons organized on different layers:

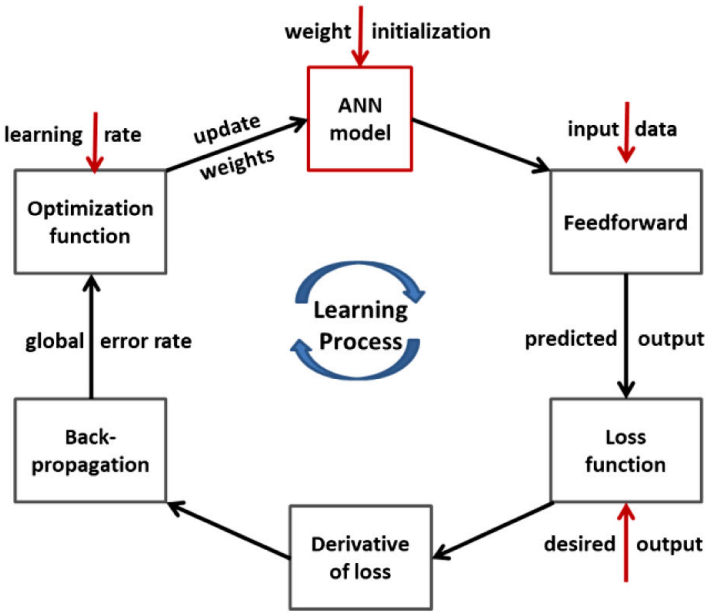


FIGURE 2.1: Supervised ANN model learning process [22]

input, *hidden* and *output* layers. At the input layer there are the neurons that receive the input data provided by the training set or the testing set; the hidden layers are designed to make a non linear transformation to the data of the input layer applying activation functions to each neuron. Activation functions permits to each node of the network to produce an output value given some input values processing the input value and mapping them with the specific type of function adopted; there exist a lot of activation functions and some of them are *exponential linear unit*, *sigmoid*, *hyperbolic tangent*, *rectified linear unit (ReLU)* and *inverse square root linear unit (ISRLU)*. Here, authors used *ReLU*.

The learning process of ANN model consists in several steps to setup an optimal configuration of the weights of the neurons, in Figure 2.1 the crucial steps to achieve this goal are illustrated.

First of all, it is necessary to *initialize* the network with random or given weights, then all the data provided at the inputs are forwarded to the whole network and the *predicted output* is obtained. At this point, a *loss function* is exploited to make an estimation on the error that occurs in between the predicted output and the *desired output*. The impact of each weight with respect to the total loss is evaluated by calculating the derivative of the loss function applied in the reverse order, i.e back-propagating from the last to the first layer of the ANN. Algorithms such as stochastic gradient descending are implemented to decrease the *global error rate* and *adjusting/updating* the weights. The process repeats iteratively and feed-forward and back-propagation keep place again to continuously balance the weights of the ANN.

The adjusting/updating weight process inherits from the error margin between the predicted output obtained at the previous iteration and the reference value. It follows

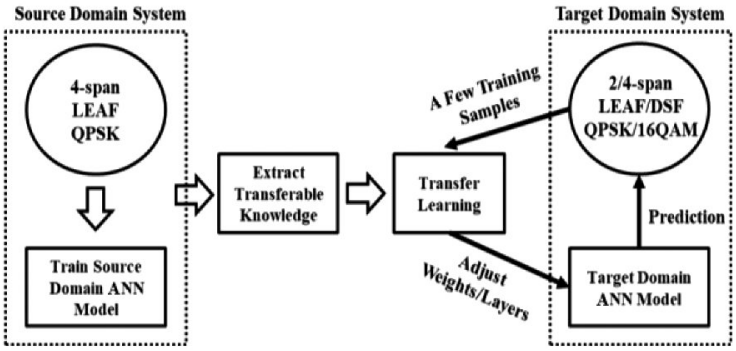


FIGURE 2.2: Transfer learning flow chart [22]

that, by providing enough reference data, it is possible to obtain an accurate model because in every cycle, the learning process upgrade its prediction capabilities. Authors show that this kind of neural network can achieve an accurate Q-factor prediction with a low root-mean-square error (RMSE).

A 4-span large effective area fiber (LEAF) with QPSK transmission is adopted as source domain, while a 4-span LEAF 16 QAM, a 2-span LEAF 16 QAM and a 3-span dispersion-shifted fiber (DSF) QPSK are used as target domains. After an initial training phase, using samples from the source domain, the training is refined using knowledge extracted from the source domain and adding a small number of samples from the three different target domains. Figure 2.2 describes the steps of the process to make predictions in the three target domains.

The adopted artificial neural network consists of one input layer, one output layer and in between three hidden layers of 120, 120, and 60 neurons. Initially, the knowledge is stored by tuning weights in the different layers where the ANN model is trained with samples of source domain; then, to perform transfer learning the weights of the hidden layer are readjusted based on a few samples from the target domain, thus speeding up the training process w.r.t re-train the network from scratch. The motivation to use the artificial neural network (ANN) as learning algorithm lies in the way which transfer learning is implemented, depending on the type of weights initialization scheme. In this case, instead of starting the learning phase from randomly initialized weights, transfer learning leverages pre-trained weights, thus enabling a fast upgrade of the ANN model with a limited amount of samples from the target domain.

The ANN model is trained using 22 separates input features which are the combinations of *output power* and *modulation format* of eleven entry channels.

In order to adapt the prediction to a real-time environment, the artificial neural network is developed in a *SDN controller*, described in their testbed implementation, that performs *online learning* to evaluate the Q-factor. Due to the lack of data, the initial training phase in the network, is executed with 16 randomly chosen channel loads. The online learning in the SDN controller allows the artificial neural network

both to make a faster collection of data and at the same time to adjust the weights of the neurons of the inner layers.

The learning algorithm stops whenever the RMSE (Root Mean Squared Error) does not decrease during two consecutive stages, thus denoting a well-suited weights setting and an acceptable prediction of the Q-factor with respect to the actual values.

Results regarding the comparison between predicted Q-factor and its actual value, show that only 20 training samples are necessary to fine-tune the weights in order to produce an accurate prediction of QoT for the three domains applying transfer knowledge. Furthermore, the choice to use artificial neural network is motivated by the good performance achieved in the prediction comparing to other methods like Support Vector Regressor and Ridge Regression.

In an extended version of their study [22], they provide more details about the design of ANN and its implementation in an optical software defined networking (SDN) controller. Authors address three aspects: the type of learned knowledge that is transferred from the source domain in the ANN model, the usefulness of this knowledge in the target domain and how much the target task is improved by using transfer learning.

The testbed implementation and data collection follows the same scheme of their previous work. Here instead, authors perform their model analysis following different methods, for instance in the first stage of modeling in which the training of the source domain keeps place, they incremented the number of homogeneous hidden layers and also compared the results varying the number of neurons in each layer. Homogeneous layers means that all layers have the same number of neurons.

In the second stage instead, the transfer training data were classified in three types based on the channel power. This is done because channel power and wavelength allocation are the most related features among the different domains.

Results for the homogeneous ANN structure model trained on the source domain show that in order to obtain an accurate model for the Q-factor prediction, 2640 training samples are sufficient. In this scenario the ANN structure is composed of three hidden layers with 128 neurons in each of them. This model is then tested with 330 test samples not previously used for training and the output is compared with different methods like Ridge Regression and Support Vector Regressor. In Figure 2.3, the accuracy of the prediction with respect to the 14 different output power levels per channel is highlighted.

The homogeneous ANN structure model for the transfer learning has to be designed taking into account that few samples from the target domain are available and, because of the limited amount of such data, finding the right configuration from the source domain plays a fundamental role. In order to conduct experiments, 20 training samples from the four span LEAF 16 QAM and the two span LEAF 16 QAM

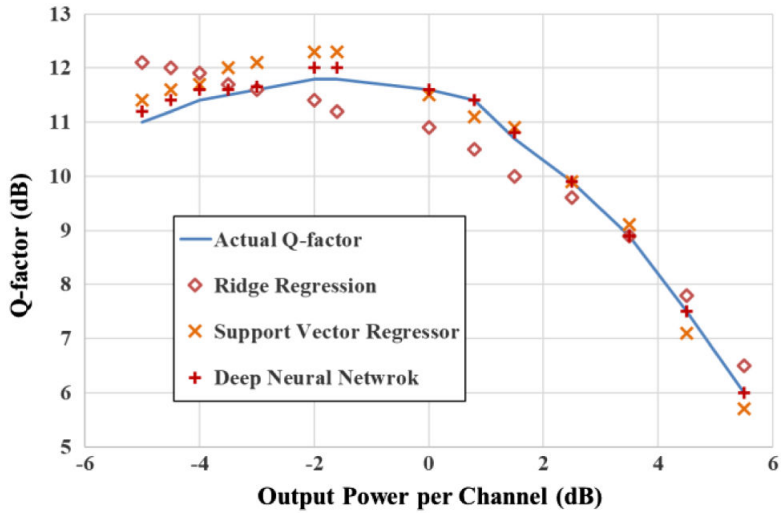


FIGURE 2.3: QoT prediction accuracy for varying launch power [22]

are selected, while 30 training samples from the three span DSF QPSK are chosen. To test the performance 1280 samples from the corresponding target domain are used and an ANN model with a number of layers in the range 2-4 and a structure of 256 neurons per layer is considered. Results show that, for the four span LEAF 16 QAM, the best accuracy is obtained considering an ANN model with two layers; for the two span LEAF 16 QAM system, the best accuracy is obtained using an ANN model with four layers and tuning the weights of the second, the third, and the fourth layer, while for the three span DSF QPSK system, the best accuracy is obtained using ANN model constituted by four layers and tuning the weights of all of them.

Performance evaluation is conducted also adding heterogeneous layers, which means, layers with different number of neurons. A two layer structure with 128 neurons in each, is used as source domain ANN model. Results, in this case, show that the highest accuracy is obtained adding 32 neurons in the third layer and tuning all the layers of the two span LEAF 16 QAM, while adding 64 neurons in the third layer and tuning all the layers of the three span DSF QPSK system.

As last step, authors compare transfer learning to a conventional training method in the training of the four span LEAF 16 QAM. In this case, results show that, when adopting transfer learning, the training target samples can be reduced from 1000 to 20, while still ensuring a reliable and accurate QoT prediction while achieving a significant reduction of time required by the training phase.

2.1.2 OSNR estimation

Optical signal to noise ratio (OSNR) is a crucial parameter to be monitored in coherent optical networks, as it measures the ratio between signal power and noise power of an optical channel. In [20], the OSNR is monitored at the physical layer to enable a suitable fast re-modeling of an OSNR predictor based on the variations of system parameters, e.g., optical launch power, chromatic dispersion (CD) and bit-rate.

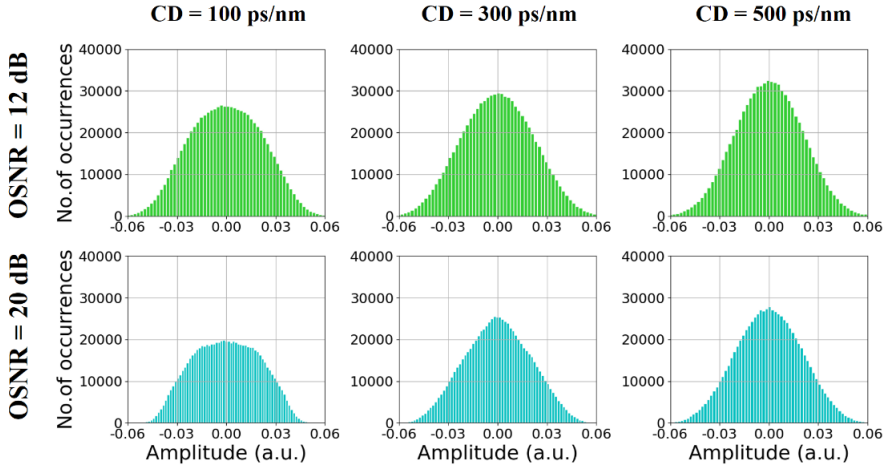


FIGURE 2.4: Morphological characteristics of amplitude histograms with same OSNR but different chromatic dispersion [20]

To speed up OSNR estimation, they proposed a transfer learning (TL) assisted deep neural network (DNN) implemented at the receiver, which is trained with numerous samples whose features are 80 amplitude histograms (AH) of the received signals plus one variance value, that reflects changes in the distribution of received data.

Authors highlight that signals with different chromatic dispersion (CD) values and amplitude histograms with same OSNRs show different morphological characteristics, as shown in Figure 2.4. This indicates that AHs based on the OSNR estimation method will depend on the residual CD, i.e it is necessary to adapt the variation of CD to enable efficient monitoring of the OSNR. Other samples are then generated changing parameters in the physical layer such as launch power, residual dispersion and bit-rate.

Transfer learning is applied by sharing values of the weights obtained from a deep neural network trained with samples from the source domain in order to adapt to the transmission parameter variation. The source DNN was trained with signals at launch power 0 dBm and the knowledge of all layers in the source model was transferred into the target model. Also in this case, the utilization of transfer learning in deep neural networks exploits the re-usage of weights in one or more layers from a pre-trained model in a new model, either keeping them fixed or fine-tuning them to ease adaptation. In Figure 2.5, the relevant steps adopted by the algorithm to perform OSNR estimation are depicted.

The DNN infrastructure is composed of seven neural layers: 81 neurons at the input layer and one neuron at the output layer; in between, there are 64, 32, 16, 8, 4 neurons, respectively. The training phase is implemented by generating dataset which contain numerous amplitude histograms (AHs), corresponding to different OSNRs at different chromatic dispersions (CDs).

Results show a superior capability of fast remodeling with respect to parameter

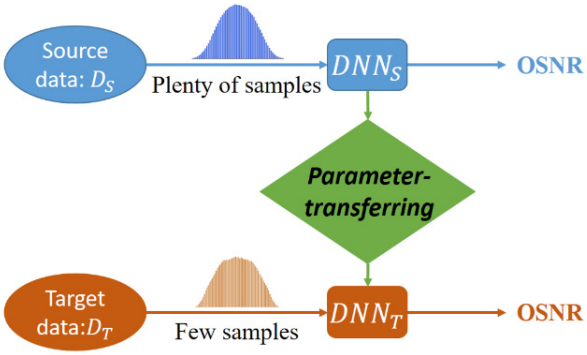


FIGURE 2.5: Transfer learning assisted DNN method for OSNR estimation [20]

system changing, helpful for real-time OSNR estimation. Moreover, both training epochs and training set sizes are reduced at the different system parameters settings (target launch power, residual chromatic dispersion and bit-rate) with respect to those obtained performing a retraining of the neural network from scratch, i.e making a training that starts from a randomly-initialized weights scheme without relying on any prior knowledge.

2.1.3 Spectrum Optimization for the Resource Reservation in Space Division Multiplexing Elastic Optical Networks

In [21] a spectrum optimization model for space division multiplexing elastic optical networks (SDM-EON) is described to enable a well-designed resource reservation algorithm for the service requests, based on transductive transfer learning.

Firstly, authors model the request R according to a six-tuple:

$$R = \{s, d, T_{arrival}, T_{start}, T_{end}, R_G, n\}$$

where the pair s and d refers to the source node and the destination node, $T_{arrival}$, T_{start} and T_{end} are the arrival time, the start time and end time of the service requests, while R_G indicates the level of the request relative to its service quality requirements. Finally, n represents the number of frequency slots that have to be reserved for that request.

As illustrated in Figure 2.6, a set of samples is collected from the source domain topology, constituted by requests of services. This set is then sorted in order to discriminate between *blocked service set*, consisting of all requests that will be blocked and not be accounted for the reservation of the network resources, *affected service set*, that instead, consists of all the requests already satisfied in the network and *spectrum de-fragmentation time set*, which contains time value of spectrum de-fragmentation. This ensemble is then used to perform the training phase and produces a so-called *pre-training* model.

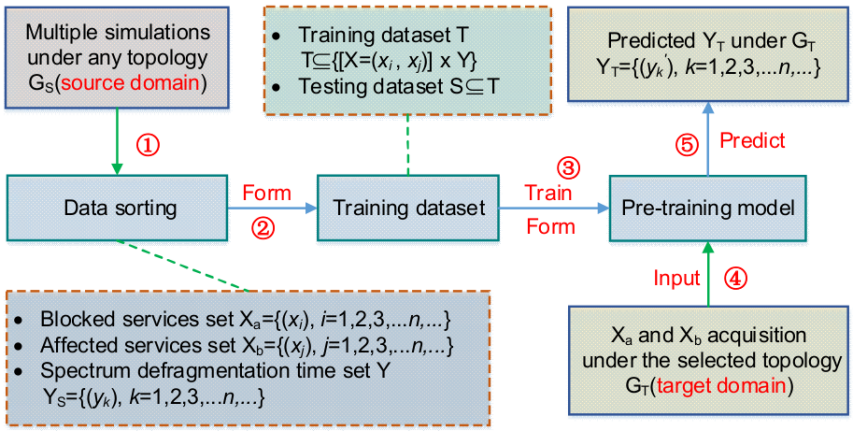


FIGURE 2.6: SMT prediction based on TTL [21]

From the selected target domain topology, the *blocked service set* and *affected service set* are used as inputs of the pre-trained model. The outputs of the model coincide with the prediction of spectrum migration time (SMT) of the requests. In practice, the *pre-trained* model represents a structure that contains the knowledge acquired in the source domain, able to predict the spectrum migration time of a specific request from the target domain. Based on this, another phase is developed that is in charge of optimizing the spectrum utilization and improve resource reservations among different services.

Authors used a six node topology as source domain, while a 14 nodes NSF topology is used as target domain. The model is trained, as said, with service requests which are organized according to a specific scheme, grouped and labeled into different kind of sets ready to be passed to the *pre-training* model.

Results highlight the possibility to decrease the blocking probability by about 67% with respect to other methods that do not take into account transductive transfer learning.

2.1.4 Comparison to related work

In this thesis, source and target domains are identified by synthetic data-sets generated with E-tool (See Section 3.4) composed of synthetic data and starting from a given network topology.

We will consider two topologies that in turn will be considered as source and target domains. The experiments will focus on the classification of lightpaths that can be or cannot be established in an optical transmission system, based on the bit-error rate (BER) value measured at the receiver.

The approaches used to conduct experiments are *feature augmentation* and the *correlation alignment*. The area under the curve (AUC) is used as metric to evaluate the performance of the classifier. The learning model adopted to perform lightpath classification are

TABLE 2.1: Predicted parameters, ML models and TL approaches in literature

Authors	PP	ML Model	TL approach
Weiyang Mo et al.[11]	Q-factor	Artificial Neural Network	Tuning-weights Feature extraction
Le Xia et al.[20]	OSNR	Deep Neural Network	Tuning-weights Feature extraction
Qiuyan Yao et al.[21]	SMT	Neural Network	Tuning-weights Feature extraction
(Us, 2019)	BER	Random Forest Support Vector Machines Logistic regression	Feature augmentation CORrelation ALignment

*PP = Predicted parameter
*ML = Machine Learning
*TL = Transfer Learning

Random Forests with 25 trees, Support Vector Machines (SVM) implementing linear kernel and then Logistic Regression.

The choice of authors of [11, 20, 21, 22] to implement a neural network based transfer learning allows them to make an accurate selection of the important features (*feature extraction*) useful to perform learning and best tune the weights of the internal layers. Indeed, their systems can leverage on a wide range of features and have to exploit the ones that best fit with their purpose.

Differently, we focus on a restricted group of features, but firstly, we had select five of them directly related to the lightpath, and then eleven ones specific of the lightpath and its neighbors, to perform the training of the classifier. (See Chapter 4)

Table 2.1 summarizes the different scenarios illustrated in the previous subsections, making a distinction between authors, predicted parameter, adopted machine learning model and transfer learning approach.

Chapter 3

Background

In the last few years ML techniques have found a lot of application fields, e.g., autonomous vehicles, search engines or facial and vocal recognition. These kinds of algorithms are designed to configure machines and adapt their behavior in order to fulfill a predetermined task. In literature, ML-based frameworks have been widely investigated and the ones leveraged in this work are *domain adaptation* and *transfer learning*. They focus on the ability of a system to make use of the knowledge obtained from data extracted from a *source domain* and enforce it to a *target domain*: i.e a scenario different from the first one, but related to it. This concept finds application in many supervised learning scenarios in which there are not sufficient samples labeled, to train a classifier in the target domain; however, it results convenient to use the small pool of them to improve knowledge learned from the source domain.

This chapter will provide some background on ML, starting from its definitions, its categories of algorithms designed to tackle different problems, its application in optical networks and finally, describing those models that are considered in this work.

The concept of Transfer Learning (TL) will be then illustrated, giving the reader the necessary tools to understand the domains of application of this technique and its implication in the performance improvements obtained in this thesis.

Finally, the chapter will discuss the procedure through which synthetic data, leveraged to conduct the experiments of this thesis, are generated, describing the *Etool* and the algorithm for the dataset generation.

3.1 Machine Learning

The term Machine Learning refers to a specific branch of Artificial Intelligence (AI). In the 1959, Arthur Samuel, the pioneer in ML, coined this term and provided the definition:

"*Field of study that gives computers the ability to learn without being explicitly programmed.*"

Indeed, two characteristics of this concept can be extrapolated: 1) it is possible to extend the application in various scenarios, automating complex tasks that normally have be executed by humans and 2) from time to time, it is possible to improve the experience of a machine to perform a specific task.

A more precise definition is proposed by Tom M. Mitchell in 1998 formalizing the learning problem in a computer program:

"*A computer program is said to learn from experience* ***E*** *with respect to some task* ***T*** *and some performance measure* ***P****, if its performance on* ***T****, as measured by* ***P****, improves with experience* ***E****.*"

It results clear from this definition that the capability of a computer program to improve the performance of a specific task is closely connected to its experience accumulated solving that task; but, *what if this experience derives from not well designed learning procedures?*

For this reason, a lot of machine learning algorithms that best fit the different problems have been developed. These algorithms can be subdivided into three main categories: *supervised learning*, *unsupervised learning* and an hybrid of the two, i.e., *semi-supervised learning*.

3.1.1 Supervised learning

In supervised learning a *vector* of **input** variables is provided to the ML algorithm and the goal is to make a prediction, as accurate as possible, of one or more output variables. In this learning method the vector of input variables is associated to either a numerical value or a well-known class to which the output variable related to this sample belongs, by the use of *labels*.

The output variable can belongs to two type of domains: if it is a **discrete variable**, then the supervised learning method exploits a *classification* problem, if it is a **continuous variable**, then supervised learning method fulfills a *regression* problem. Both of them have the aim to find a *mapping function* called *hypothesis*, that is able to estimate from an input vector, an unknown output value, also called *target value*.

In practice, supervised learning tries to extract the common properties from the input data and make predictions based on that. The problem can be formalized as follow:

Let *n_features* be the number of features and be *n_sample* the number of samples. We define a feature vector $x \in \mathbb{R}^{n\_feature}$ and a set of labels y such that:

$$\begin{cases} y \in \{-1, 1\}, & \rightarrow \textbf{(binary) classification} \text{ problem} \\ y \in \mathbb{R}, & \rightarrow \textbf{regression} \text{ problem} \end{cases} \tag{3.1}$$

Given a collection of input variables $X = \{(x_i, y_i) \mid 0 \leq i \leq n\_samples\}$, the goal is to learn a mapping function $y* = f(x)$, such that $y* = f(x) + e \mid e \simeq 0$, where e is the *prediction error*.

Usually this methods require a lot of data in order to become efficient and reach an high level of accuracy of predictions.

A collection of vectors of input variables which contains samples that are *labeled* composes the *training set* if it is used to perform the training step of the learning algorithm, then a separate *testing set* it is used to test the capabilities of the learning algorithm.

Supervised learning can be subdivided into two main classes: *parametric models* and *non parametric models*.

In parametric models a fixed set of parameters w is estimated. After the training, the prediction associated to the new inputs is performed using only the parameters that have already been learned and the training data can be discarded. The simplest parametric models for regression and classification are linear models, i.e., a linear combination of nonlinear basis functions. Basis functions can be of different types, for example, *gaussian*, *sigmoidal*, *polynomial*. A set of basis functions can be applied for each component in the case of multiple output variables. The application of linear models is usually limited to problems that have a input space with few dimensions.

In nonparametric models, the number of parameters is dependent on the set size of the training. Some examples of approach that exploit nonparametric models are K-Nearest Neighbors (KNNs) and Support Vector Machines (SVMs). These models can be used in order to solve both *classification* and *regression* problems.

3.1.2 Unsupervised learning

Unsupervised learning tackles scenarios where the class of the samples data used to train a model is not known. In practice, the *feature vector* is not associated to the *label*. This kind of learning models are useful to recognize the different classes based on the data that are passed to the model during the training phase.

Unsupervised learning finds application in the vast majority of fields, especially because in reality there exist much more scenarios in which the output variable cannot be known in advance; *clustering* problems and *cluster analysis* are the most common applications. Clustering refers to the identification of groups in the pool of data that share similar characteristics. The similarity is evaluated taking into account the type of data and in particular the features vector selected to train the learning model; it is typically described by a distance function. The resolution of this task points to group data into clusters, in which the *intra-cluster* similarity is high, while *inter-cluster* similarity is low.

Examples of clustering types are *hierarchical clustering*, *K-means clustering*, *K-Nearest Neighbors (KNNs)*, and *Principal Component Analysis (PCA)*.

As it will be described in Subsection 3.3.2, we make use of an unsupervised DA technique, called CORAL (CORrelation ALignment), which exploits the capacity to approach as long as possible target domain and source domain, through a geometric transformation of data. We assume that our target domain dataset contains unlabeled samples. By this fact, we can exploit the totality of target domain samples to perform the transformation step and improve learning from the input data.

3.1.3 Supervised learning Algorithms

Random Forest

Random Forest (RF) is a simple algorithm that can be used in *classification* problems and *regression* problems. A RF classifier is a meta estimator that creates a forest of *decision tree classifiers* and determines the class most often chosen by the trees.

In order to create the forest, each tree of the ensemble is built following the *bagging* technique. The idea behind the bagging algorithm is to create B datasets, sampling with replacement I instances from the original training set. N identifies the number of instances of the initial training set, this means that bagging will produce B trees with I samples in each. The $B_i, i = 1, 2, ..., B$ tree produces an estimation of the class C_i. The final predicted class C is the one exhibiting *most occurrences*.

The randomness introduced by this method is very useful because it reduces the variance of the forest estimator w.r.t the individual decision tree which tends to show high variance and lead to *overfitting* problem. RF algorithms achieve low variance by the combination of the inner trees and reach high performance in the classification.

In this thesis a random-forest classifier which predicts the probability for unestablished lightpaths to exceed a given BER threshold is used. Based on work exposed in [16], we decide to use 25 estimators in order to preserve a good tradeoff between achieved performance and computational time to complete the training and the testing phases.

Support Vector Machines

Support Vector Machine (SVM) is a ML supervised algorithm that locates each sample of the dataset into a geometric space of n dimensions, where n is the number of attributes (or features) available. SVMs models try to draw into the n-dimensional space an hyperplane that separates or divides data belonging to different classes. It is used in both *classification* and *regression* problems.

Consider a group of data that have been classified into two classes; if it is possible to linearly separate the two groups of data points, the two classes are *separable* and SVM can draw an hyperplane of n dimensions. In the example reported in Figure 3.1

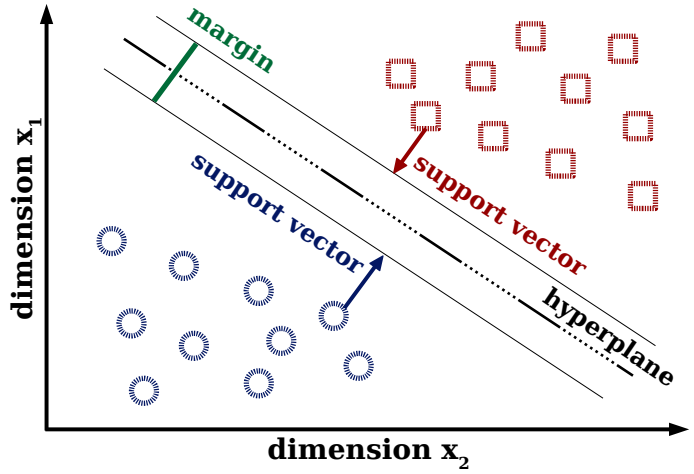


FIGURE 3.1: Support Vector Machines in two dimensions and with two data points classes

n is equal to *two*, because only two features are present, so the hyperplane coincides with a straight line.

If two classes are linearly separable, there exist an infinite number of hyperplanes that could be drawn; SVM optimize this choice selecting the one that maximizes the separation among the two classes and uses it as *decision boundary*. Figure 3.1 represents a particular scenario in which a training set with two classes is used to train the SVM algorithm. The separation between the two classes is said *margin*, while the points which are closest to the upper and lower margin of the hyperplane are called *support vectors*.

SVM algorithms are the extension of linear models such as, e.g., *linear regression* and *logistic regression*, indeed they can shift on n different dimensions to identify the best decision boundary based on the number of attributes and the distributions of the training set data. Attributes of a sample identify the geometric space of data, but generally data are not separable in a low dimensional space, while they may became separable in a space with more dimensions.

SVM is considered a *kernel method*: indeed, it exploits different types of kernels to adapt also to non linear models. With the use of different *kernel functions*, such as, e.g., linear kernel, polynomial kernel, gaussian or radial basis function (RBF) kernel, it is possible to abstract in higher dimensions the sample features mapped at a lower dimensional space. The general definition of the kernel is $K(x, y) = \langle \phi(x), \phi(y) \rangle$, where $\phi(x)$ and $\phi(y)$ are the mapping feature function from a n dimensional space to an m dimensional space, with $m > n$.

In this thesis a *linear kernel* will be used,

$$K(x_i, y_i) = x_i \cdot y_i$$

it is the simplest one, and it receives two vectors in input and computes the scalar product of the two, as it has been proved to yield good classification results and limits the overfitting problem; future investigations of this work can be done considering the adoption of a different kernel type.

SVM algorithms, in general, lead to accurate predictions, but may require significant computational time if the number of considered dimensions is high. Like RF and ANN, SVMs find application in optical networks especially in *failure prediction* and *identification of the cause at the root* [14].

Logistic Regression

A linear regression model can be generalized to a classification model, in our case a binary classification, selecting a *Bernoulli* distribution, because it is more suitable to indicate a binary response $y^* \in \{0, 1\}$ [12].

The selection can be described formally with:

$$p(y^* \mid x, \beta) = Ber(y^* \mid \mu(x)) \tag{3.2}$$

where, x is the features vector, β denotes the regression weights and $\mu(x)$ the mean value of x, which can be written as $\mu(x) = \mathbb{E}[y^* \mid x] = p(y^* = 1 \mid x)$.

The computation of the linear combination of the inputs is performed additionally passing through a function which ensures that $0 \leq \mu(x) \leq 1$ through the use of the *basis function* $sigm(\eta)$, i.e., the *sigmoid* function, also called *logistic* or *logit*.

$$\mu(x) = sigm(\beta^T x) \tag{3.3}$$

The sigmoid function is defined as:

$$sigm(\eta) \stackrel{\text{def}}{=} \frac{1}{1 + e^{-\eta}} = \frac{e^{\eta}}{e^{\eta} + 1} \tag{3.4}$$

and its shape is represented in Figure 3.2, where $sigm(-\infty) = 0$, $sigm(0) = 0.5$ and $sigm(+\infty) = 1$. Observing the trend of the sigmoid function it is possible to note that the response value $y*$ is equal to 0 when $sigm(\eta) < 0.5$ or equal to 1 when $sigm(\eta) \geq 0.5$. The *decision rule* for our response value $y*$ is determined by the threshold set when $\eta = 0 \stackrel{i.e}{\Rightarrow} sigm(0)$.

Substituting the $\mu(x)$ equality of Equations 3.3 into the 3.2, we obtain the formal definition of *Logistic Regression*:

$$p(y^* \mid x, \beta) = Ber(y^* \mid sigm(\beta^T x)) \tag{3.5}$$

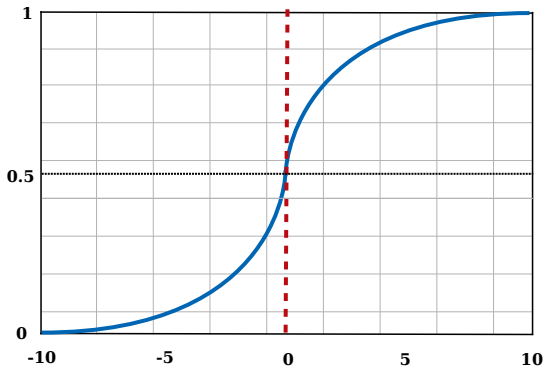


FIGURE 3.2: Sigmoid function - Decision rule in red, in this case the threshold value is 0.5.

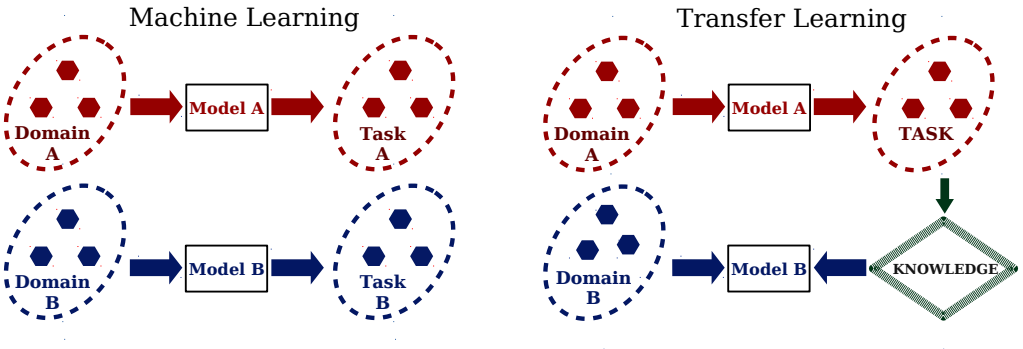


FIGURE 3.3: Traditional ML-Training and testing of the model on samples of the same domain & TL process-Knowledge of a model transferred and applied in other related domain

Despite the name, this method does not tackle a regression problem, but rather a classification one.

3.2 Transfer Learning

If ML approaches typically aim to the training of a model with data of a particular domain and evaluate its performance in resolving a particular task within the same domain, TL tries to use the knowledge obtained in a particular scenario and transfer it to another related one, thus allowing to train a model within one domain and apply it to a different one during the test phase. In Figure 3.3 the main differences among the two approaches are shown. In human learning mechanisms, a lot of examples dealing with *transfer learning* are present, e.g., the knowledge of a color results useful to understand and make a distinction with another one, the idea of a cat can be helpful to recognize dogs. TL aims to extract the knowledge, acquired by a *source domain* data trained model, to solve a *source task* and apply it to another *target task* related to *target domain* data [15]. A fundamental motivation to investigate TL is that, when applying ML models trained on a given domain to a different one, it is

needed to store and upgrade the learned knowledge without retraining the model from scratch, in order to save time and speed up the following processes.

In order to implement a particular transfer learning approach, it is important to correctly define the domains of application and their characteristics, i.e., features space and distributions which are associated to them.

The majority of the proposed approaches present in literature, including ours, start from the definitions of *domain* and *task*:

- a *domain* D is identified by two components: a feature space χ and a marginal probability distribution $P(X)$, where $X = \{x_1, x_2, x_3, ..., x_n\} \in \chi$;
  χ defines the space of all features, while X is the sample composed by the x_i, with $i = 1, 2, \ldots, n$, features.
  In general, a domain is represented by $D = \{\chi, P(X)\}$ and it follows that: $D_1 \neq D_2$, if $\chi_1 \neq \chi_2 \vee P_1(X) \neq P_2(X)$;
- a *task* T consists of two components: a label space ν and an objective predictive function $f(\cdot)$. The former identifies all the possible classes y_i, with $i = 2, 3, \ldots$ to which a sample can be associated to, so if it is a binary classification we can have y_1 =True and y_2 =False; the latter is used to predict the label y of an unknown sample X, i.e., $f(X) = P(y \mid X)$, a task is represented by the notation $T = \{\nu, f(\cdot)\}$.

After the definition of domain and task, it is necessary to focus on the discrimination among the two main environments in which TL is really applied: *source* and *target* domains.

- Let D_S be the source domain data-set denoted by

  $$D_S = \{(x_{S_1}, y_{S_1}), \ldots, (x_{S_s}, y_{S_s})\}$$

  where $x_{S_i} \in \chi_S$ is the $i-th$ instance of features vectors collection χ_S and $y_{S_i} \in \nu_S$ is the $i-th$ class label, with $i = 1, 2, \ldots, s$;
- Let T_S be the learning task associated to D_S;
- Let D_t be the target domain data-set denoted by

  $$D_T = \{(x_{T_1}, y_{T_1}), \ldots, (x_{T_t}, y_{T_t})\}$$

  where $x_{T_i} \in \chi_T$ is the $i-th$ instance of features vectors collection χ_T and $y_{T_i} \in \nu_T$ is the $i-th$ class label corresponding to the output, with $i = 1, 2, \ldots, t$;
- Let T_T be the learning task associated to D_T.

Note that it is very common that $0 < t \ll s$, which states that the number of samples of the target domain is very limited with respect to that of the source domain. Now

TABLE 3.1: ML and TL Learning settings

	Setting	D_S and D_T	T_S and T_T
Machine Learning	Traditional	$D_S = D_T$	$T_S = T_T$
Transfer Learning	ITL	$D_S = D_T$	$T_S \neq T_T$
	UTL	$D_S \neq D_T$	$T_S \neq T_T$
	TTL	$D_S \neq D_T$	$T_S = T_T$

*ITL = Inductive Transfer Learning
*UTL = Unsupervised Transfer Learning
*TTL = Transductive Transfer Learning

that the basic definitions and assumptions have been discussed, a general definition of TL concept [15] can be provided:

Transfer learning definition: Given D_S and T_S, D_T and T_T, TL aims to increase the effectiveness of the target learning predictive function $f_T(\cdot)$ in D_T exploiting the knowledge of D_S and T_S, where $D_S \neq D_T$ or $T_S \neq T_T$.

Note that in the particular scenario where $D_S = D_T$ and $T_S = T_T$ the learning problem coincides with a traditional ML problem. Based on the above definition, authors in [15] provide a taxonomy of the different types of TL approaches, depending on the characterization of elements in the *source* and in the *target* domains and tasks. In particular, they identify three types of TL: *inductive transfer learning*, *transductive transfer learning* and *unsupervised transfer learning*. Table 3.1 highlights the main commonalities and different aspects of the above mentioned TL settings.

In *inductive TL*, the tasks achieved in source and target domains are different, while domains are the same, formally $D_S = D_T$ and $T_S \neq T_T$. In *unsupervised TL*, similarly to the previous one, source and target tasks differs even if they are related: it focuses on solving clustering and estimation of density problems in the target domains, and no labeled data are available in both domains. In *transductive TL*, source and target domains are different, while source and target tasks are equal, formally $D_S \neq D_T$ and $T_S = T_T$.

In this thesis experiments are designed to consider two domains, with same feature spaces χ_S and χ_T, but different marginal probability distributions $P_S(X)$ and $P_T(X)$, and they are used to train a learning model able to execute the same task in the source and in the target domain ($T_S = T_T$), i.e., a TTL setting has been considered.

3.3 TL approaches

The overview provided in Chapter 2 discusses the TL approaches that have been adopted in optical networks and highlights the importance of neural networks (NNs) and *weight tuning techniques* in order to transfer knowledge through the inner layers of the networks. Differently, our approach is to explore different *domain adaptation*

techniques in the development of algorithms that can be exported from a domain to another one. In this sense, DA can be considered as a sub category of the TL methodologies. More in detail, we start from the assumption that the size of the training dataset gathered from the target domain is limited, so we have to combine the large pool of source domain samples with the few samples of the target domain. Practically, our analysis focus on the *transformation* of the source and target feature spaces χ_S, χ_T and then, it tries to find useful training strategies for our classifier.

The following section describes the two approaches, Feature augmentation [10] and Correlation Alignment (CORAL) [19], used to perform the transformation of our feature spaces, motivating the choice and illustrating their advantages and disadvantages.

3.3.1 Feature augmentation

As said, the task of DA consists in the exportation of a trained learning algorithm from a domain to an other one; this is not so easy due to the lack of labeled target data useful to extract precious information from the application domain (target domain). The technique from which we take inspiration is based on the transformation of a DA learning problem into a standard supervised problem in which our classifier can be applied.

Feature Augmentation (FA) [10] transforms the feature space of source and target domains by augmenting it. The result of this transformation is then used to train our classifier. As the authors of [10], we also focus on the fully supervised case, i.e., a scenario in which data from source and target domains are entirely labeled. This option has been chosen because we generate data synthetically, so we assumed that a long period of network monitoring in the various source and target domains has been done and there is the availability of labeled data.

Problem formalization

In order to describe the steps to compute the transformation, we first introduce some notations. Given:

- Input space χ;
- Output space ν;
- Source domain D_S;
- Target domain D_T;

For the sake of easiness we assume $\chi = \mathbb{R}^F$, where $F > 0$. Actually F is the cardinality of the feature space. What we want to obtain is the design of the *augmented space* of input features and the mapping functions among source and target data.

The augmented feature space can be defined as:

$$\check{\chi} = \mathbb{R}^{(K+1)F}$$

where K is the number of domains we consider, while the mapping functions of source and target data are respectively, ϕ_s, ϕ_t and defined as:

$$\phi_s, \phi_t : \chi \rightarrow \check{\chi}$$

$$\phi_s(x) = \langle x, x, 0 \rangle \tag{3.6}$$

$$\phi_t(x) = \langle x, 0, x \rangle \tag{3.7}$$

where $\mathbf{0} = \langle 0, 0, \ldots, 0 \rangle \in \mathbb{R}^F$ and it corresponds to the zero vector.

The equations 3.6 and 3.7 correspond to the linear transformation of the feature space and introduce a replication of the features with respect to their proper domain. To clarify this concept, we try to map the problem to our application scenario. Given two domains ($K = 2$), as in our case they are two network topologies, we expand their feature spaces of $K + 1$, translating $\chi = \mathbb{R}^F \rightarrow \tilde{\chi} = \mathbb{R}^{3F}$. For each expansion $\tilde{\chi}$ there are $K + 1$ replicas of the same features. The first replica refers to the *general version* of the features, the second is associated to the *source-specific version* and the third matches with the *target-specific version*.

In the training phase, samples that belong to the source domain set to *zero* the vector of the *target-specific version*; on the contrary, samples that come from the target domain *reset to zero* the vector corresponding to the *source-specific version*. Note that, during the training phase, two versions of the same feature are always seen by the classifier, what changes is the distribution of the weights into the feature vector that is provided to the learning algorithm and the trade-off between source, target and general version is regulated by the supervised learning algorithm.

Authors used the example of *Hilbert space χ with kernel* $K : \chi \times \chi \rightarrow R$. The kernel K can be written as the dot product of two vectors in χ: $K(x, x') = \langle \phi(x), \phi(x') \rangle_\chi$. Defining again ϕ_s and ϕ_t:

$$\phi_s(x) = \langle \phi(x), \phi(x), 0 \rangle \tag{3.8}$$

$$\phi_t(x) = \langle \phi(x), 0, \phi(x) \rangle \tag{3.9}$$

and compute the expanded Reproducing Kernel Hilbert Space (RKHS) using the original kernel:

$$\check{K}(x, x') = \begin{cases} 2 * K(x, x') & \rightarrow \textbf{same domain} \\ K(x, x') & \rightarrow \textbf{different domain} \end{cases} \tag{3.10}$$

So considering the kernel as metric to evaluate the similarity of data we can conclude that samples belonging to the same domain are naturally two times more similar

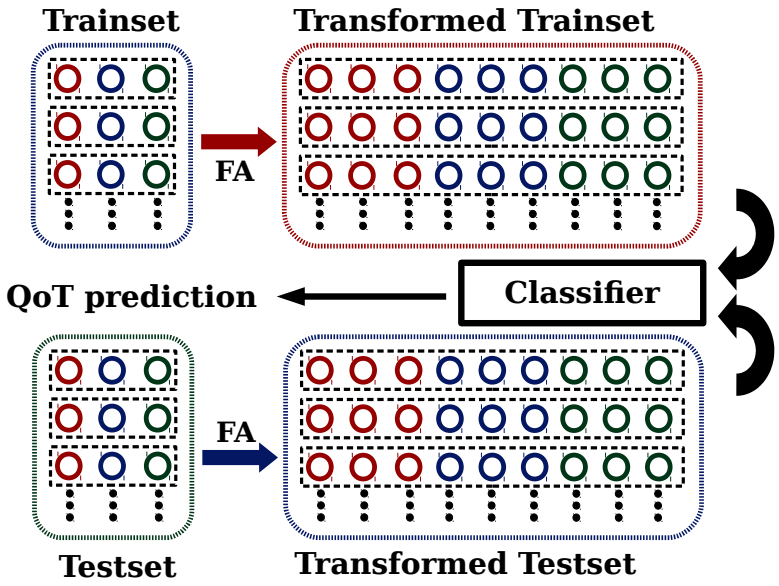


FIGURE 3.4: Feature Augmentation (FA) - General scheme

than samples that instead, belong to different domains. For this reason, the influence provided by instances of the *target domain* is twice higher than influence provided by source instances during the testing phase.

Note that augmentation transformation process is applied to all samples, both in the training and test phases. Figure 3.4 represents the general scheme followed by data before *training* and *testing* phase.

3.3.2 CORrelation ALignment

Typically, supervised ML algorithms perform well if a large set of labeled samples are provided in input and if the model is evaluated using samples test which follow the same distribution of the training data. The domain shift leads to performance degradation due to diversity among source and target data distributions. In the previous approach [10], a fully supervised DA algorithm has been developed, that replicates the features space and uses it to feed the learning algorithm. But frequently, there is no availability of labeled target data and for this reason it is impossible to adopt a supervised TL approach.

This section describes a different DA technique, called CORAL [19], which minimizes domain shift by aligning the input feature distributions of source and target datasets. In practice, authors propose a method that exploits the analysis of the second-order statistics, i.e., the *covariance* of the features in the two domains and applies a *linear transformation* to the features of the source domain.

Formally, it is an *unsupervised learning* DA technique, indeed, the target data are not labeled and are used to re-color the distribution of the source data, but after the

transformation, *supervised learning* trains the classifier with transformed source data. In our work this approach has been developed because it allowed us to leverage a large dataset of unlabeled data from the target domain to compute the transformation of the source data.

Problem formalization

Given a source domain $D_s = \{\vec{x}_i\}, \vec{x} \in \mathbb{R}^D$ labeled with labels $L_s = \{y_i\}, y \in 1, \ldots, L$, and a target domain $D_t = \{\vec{z}_i\}, \vec{z} \in \mathbb{R}^D$. The vectors $\vec{x}$ and $\vec{z}$ represent the feature vectors provided at the input with dimension D. The feature vectors undergo to a prior *normalization* step, that sets to *zero mean* and *unit variance* the distribution of the data:

- **Mean** of the *source* feature vector: $\mu_s = 0$;
- **Variance** of the *source* feature vector: $\sigma_s = 1$;
- **Mean** of the *target* feature vector: $\mu_t = 0$;
- **Variance** of the *target* feature vector: $\sigma_t = 1$;

The goal is to minimize the distance between the distributions of the source and target features, calculating the covariance and applying a *linear transformation* P to source features and using *Frobenius norm* as the metric to evaluate the distance between matrices:

$$\min_P \|C_{\hat{s}} - C_t\|_F^2 = \min_P \|P^T C_s P - C_t\|_F^2 \tag{3.11}$$

where $C_{\hat{s}}$ is the covariance matrix of the transformed source domain, C_s is the covariance matrix of the original source features, C_t corresponds to the covariance matrix of the target features, and $\|\cdot\|_F^2$ is the *Frobenius norm*. Authors identify an optimal solution of the transformation P, [19]; it corresponds to:

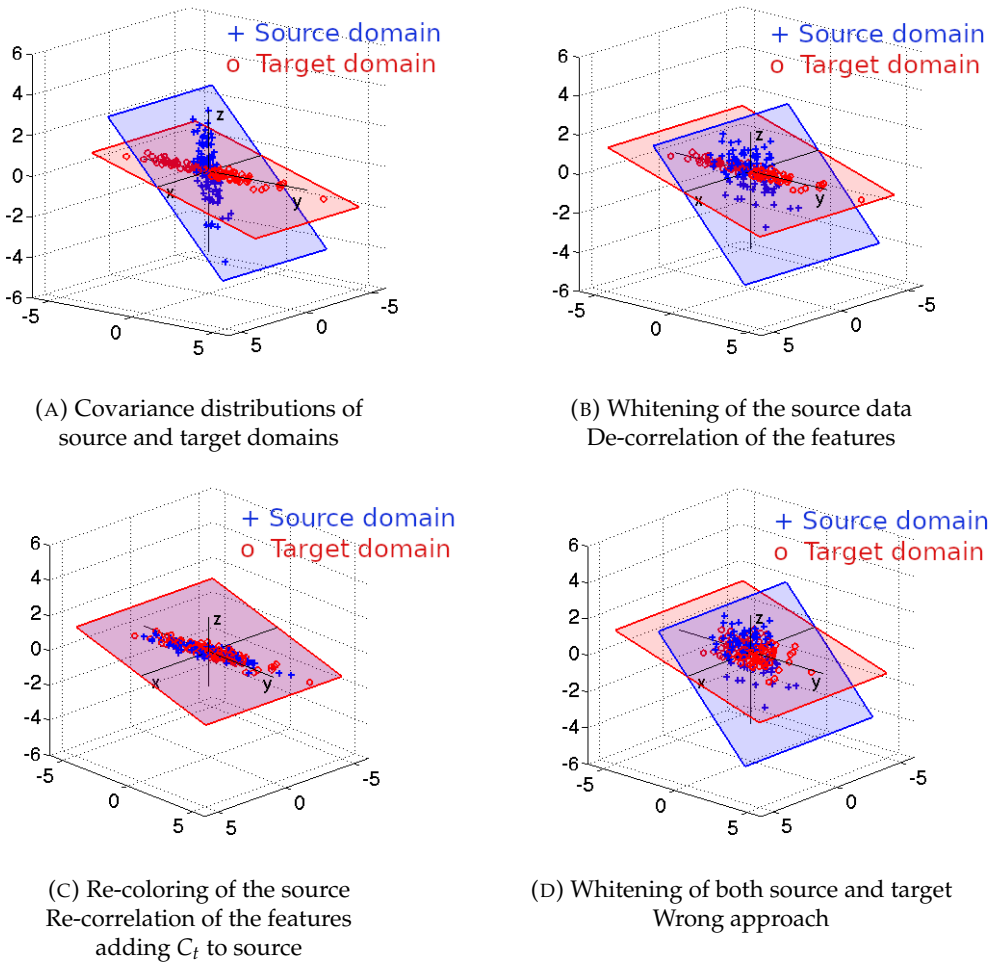
$$P = \underbrace{(U_s \Sigma_s^{\frac{1}{2}} U_s^T)}_{W} \underbrace{(U_t \Sigma_t^{\frac{1}{2}} U_t^T)}_{C} \tag{3.12}$$

where, **W** and **C** correspond to Singular Value Decomposition (SVD) conducted on C_s and C_t. In particular, **W** identifies the procedure of *whitening the source data,* $D_s \xrightarrow{W} \hat{D}_s$, while **C**, that of *re-coloring transformed source ($\hat{D}_s$), with covariance matrix of the target data* C_t, $\hat{D}_s \xrightarrow{C} \breve{D}_s$. The whitening procedure (W) of the source domain is done performing the following operation:

$$\hat{D}_s = D_s * C_s^{-\frac{1}{2}} \tag{3.13}$$

The re-coloring procedure (C) of the whitened source with target covariance feature is executed by calculating:

$$\breve{D}_s* = \hat{D}_s * C_t^{\frac{1}{2}} \tag{3.14}$$



(A) Covariance distributions of source and target domains

(B) Whitening of the source data De-correlation of the features

(C) Re-coloring of the source Re-correlation of the features adding C_t to source

(D) Whitening of both source and target Wrong approach

FIGURE 3.5: CORrelation ALignment (CORAL)[19]

Then the transformed source domain data is used to train the classifier with supervised learning methods.

Figure 3.5 illustrates different cases of two data points distributions belonging to a *source* and to a *target* domain.

Figure 3.5a shows how the covariance distributions of the two data appear before applying the linear transformation. The transfer of the knowledge of a trained classifier between the two domains is crucial due to the diversity of their spatial covariance distributions. After applying the *whitening* of the source (Figure 3.5b), data *rotate* and distribute in a subspace of the previous one, whereas nothing happens to target data. In Figure 3.5c, the re-coloring phase takes place and the covariance matrix of the target features C_t is added to the *whitened* source; it results a scenario in which target and source distributions are well aligned. In this scenario, the *transformed* feature space can be used to train the classifier. Figure 3.5d depicts a particular case in which whitening has been applied to data belonging to both source and target domains. This causes a rotation both of source and target data and consequentially the creation of two different sub-spaces on which data reside, that causes the failure

of this method. Other approaches can be performed, i.e., the *whitening* of the target features and then *re-coloring* with source data; it has been proved that this method obtains lower performance with respect to the other one [8, 6], and so we decided to develop the first approach.

3.4 BER calculation - The E-tool

The generation of data is made synthetically by simulating BER measurements from the field. We adopt the generation tool described in [16] (*E-tool*), which is able to produce an estimation of the *pre-FEC BER* of a candidate lightpath given a particular modulation format, i.e., the bit error rate (BER) at the *input* of the forward error correction (FEC) soft decoder.

The majority of FEC codes base their selection on a *threshold value*, i.e., if the pre-FEC BER is lower than a determined value, then the *output* of the FEC will be able to satysfy BER system requirement, with high probability. Given that, there exists continuity in between the pre-FEC and the BER requirement of the system, it has been considered pre-FEC as *target* value of the BER and a typical value of BER target is given by $T = 4 \cdot 10^{-3}$ [2].

In optical system affected only by chromatic dispersion (CD) and additive white Gaussian noise (AWGN), the pre-FEC BER depends on the pre-FEC signal to noise ratio (SNR). So, fixing the BER target value, it is possible to compute the required SNR.

A good link budget estimation that considers launch power, gains and losses of the optical signal allows the estimation of pre-FEC SNR and, if it exceeds the required SNR, then lightpath can be established.

Although nonlinear propagation effects can appear in the optical signal, it has been assumed the scenario where the system behaves like a linear one and nonlinear propagation effects are treated as independent contribution of AWGN with a proper power signal: the *non linear interference power* P_{NLI}. P_{NLI} depends on the input power and modulation format of each channel [3, 5], and can be computed starting from the calculation of the input power (P_{in}) of a standard single-mode fiber (SMF).

The following formula has been adopted to obtain a conservative value of P_{in} and it is based on the assumption made in [2], about the Gaussian modulation format:

$$P_{in} = \frac{G - 22}{3} \text{ [dBmW]} \tag{3.15}$$

where G is the gain that we consider of 20 dB, carried by optical amplifiers, located 100 Km far away one from the other, that restore the optical signal power.

After the computation of P_{in}, a conservative value of P_{NLI} can be obtained considering the channel bandwidth of the nearest neighbors and the modulation formats; the numerical value of P_{NLI} is found following the approach of [5]. Then P_{NLI} is converted to a loss term and added to the link budget.

Other penalty terms are defined and added to the link budget by the E-tool: *back-to-back penalties* in a 37.5 GHz flexible grid network for BPSK, QPSK and 8QAM, 16QAM, 32QAM and 64QAM modulation formats; values kept from [**Table I**, 2] and [**Fig. 7**, 23] and a *system margin parameter* randomly extracted from an exponential distribution with average 2 dB, that is associated to fast-varying penalties.

Chapter 4

Transfer Learning framework for QoT estimation

This chapter provides a detailed description of the transfer learning framework built in order to perform our experiments. The framework required by each TL approach will then be illustrated in details.

Finally, we present the design of experiments and discuss the generation process of the train and the test datasets. Note that the development of the complete TL framework is written in Python programming language, with the use of libraries such as *scikit-learn* to implement all the learning algorithms.

4.1 QoT prediction framework

In this thesis, we adopt the classifier proposed in [17], which is trained with samples belonging to a training set gathered from a specific domain and then it is tested with samples of a test set acquired independently from the training set (see Figure 4.1).

Each sample is composed by a variety of features, that characterize the different lightpaths. More in detail, we identify 11 features for each lightpath:

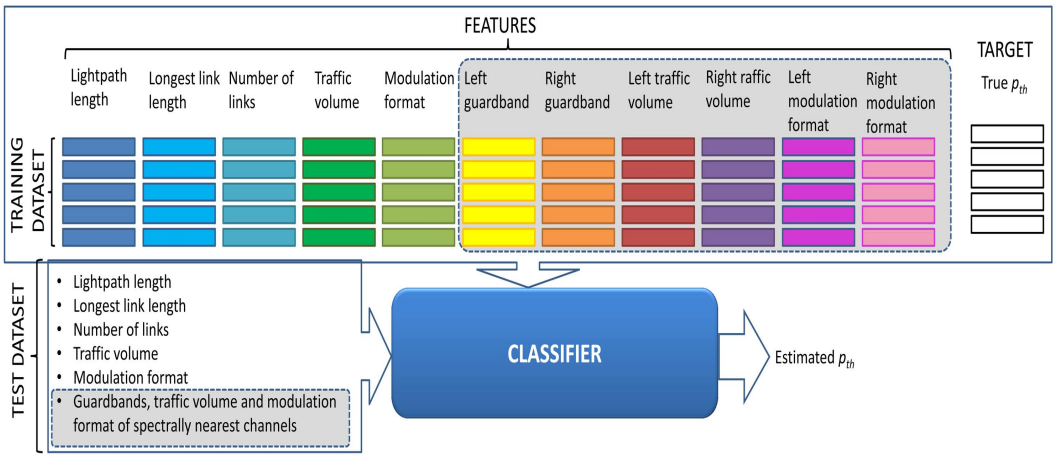


FIGURE 4.1: Classification process for the QoT estimation [17]

- Lightpath length;
- Longest link length;
- Number of traversed links;
- Traffic volume;
- Modulation format;
- Left & Right guardband;
- Traffic volume of the left and right nearest optical channel;
- Modulation format of the left and right nearest optical channel.

The last six attributes are optional and characterize the left or right neighbor channels of the lightpath, identified according to their spectral allocation, and are expected to capture the effects of cross-channel nonlinear interference.

We provide at the input of our classifier different routes with different combinations of attributes and for each of them the classifier returns an output probability p_{th} that the lightpath configuration will exceed a given *threshold* value th of the BER measured at the receiver.

The predicted class (True or False) is then obtained according to the following rule:

$$\begin{cases} \textbf{True} & \text{if } \; p_{th} < th \\ \textbf{False} & \text{otherwise} \end{cases} \tag{4.1}$$

Note that different choices of this value can affect the performance of the classifier; i.e., increasing the value of th decreases the number of samples that we classify as over threshold (positives) and, on the contrary, the number of lightpaths classified as below threshold (i.e., negative) increases.

4.2 Transfer Learning Framework

In order to apply the TL methods presented in Section 3.3, we consider two different topologies. One represents the source domain (D_s), the other represents the target domain (D_t).

Each domain is associated to a source training dataset $\mathcal{R}_{source}$, a target training dataset $\mathcal{R}_{target}$ and a separate target test dataset $\mathcal{E}_{target}$.

Every dataset contains a set of lightpath samples labeled according to a binary classification, i.e., *True* or *False*, depending on their associated BER value, computed as described in 3.4.

Basically, we design our experiment with two different *data sampling* approaches:

- *only variable target sampling*: it means that we do not perform data sampling from the source training dataset $\mathcal{R}_{source}$, but we only extract samples randomly from the target training dataset $\mathcal{R}_{target}$;
- *fixed source sampling plus variable target sampling*: it means that we consider the entire source training dataset $\mathcal{R}_{source}$ and combine it with a variable data sampling from the target training dataset $\mathcal{R}_{target}$.

In the following subsections, we describe the five different TL *settings* considered in the experiments: *only source testbed*, *only target testbed*, *mixing testbed*, *Feature Augmentation testbed* and *CORAL testbed*, highlighting the sampling rules adopted in each setting for the construction of the training sets $S \subseteq \mathcal{R}_{source}$ and $T \subseteq \mathcal{R}_{target}$.

S and T subsets are then merged together in order to produce the dataset to give as the input to our classifier.

At the end of each testbed subsection a formal description of the data sampling relative to each setting will be provided together with a simple graphical scheme to let the reader easily understand the implementation steps to build our framework.

4.2.1 Only source testbed

In this setting we consider a subset $S \subseteq \mathcal{R}_{source}$ of variable size obtained via random sampling, whereas $T = \emptyset$.

This scenario has been considered to inspect the performance of our classifier when it is trained only with samples belonging to training set $\mathcal{R}_{source}$.

Figure 4.2 illustrates this setting, while below, the sampling process from the $\mathcal{R}_{source}$ and $\mathcal{R}_{target}$ training set, which allows to obtain the data useful to perform the experiment, is described formally

$$S + T \;\; s.t \begin{cases} \mathcal{R}_{source} & \xrightarrow{random\ sampling} \;\; S \neq \emptyset \\ \mathcal{R}_{target} & \xrightarrow{zero\ sampling} \;\; T = \emptyset \end{cases} \tag{4.2}$$

4.2.2 Only target testbed

In this second setting, we consider a subset $T \subseteq \mathcal{R}_{target}$ of variable size obtained via random sampling, whereas $S = \emptyset$.

This setting identifies the cases in which the *training phase* is conducted exploiting only samples that belong to training set $\mathcal{R}_{target}$.

Through this scenario we want to analyze the behavior of our classifier and its ability to predict the classes of samples belonging to a given domain, when trained with data belonging to the same domain.

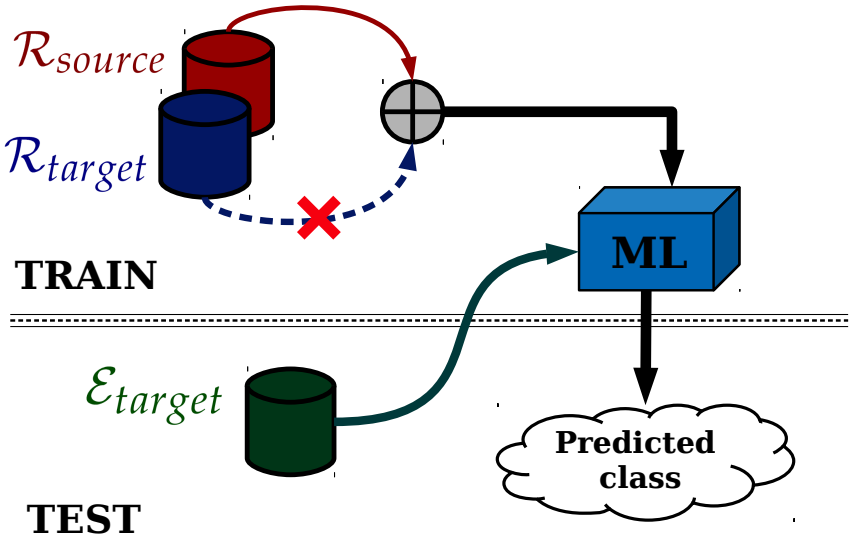


FIGURE 4.2: Only source testbed - Variable number of samples extracted from the source domain training set and zero samples extracted from the target domain training set

In practice, this is the only setting in which we perform traditional ML algorithm, indeed it is clear that $D_s = D_t$ and $T_s = T_t$, i.e., domains and tasks remain unvaried.

The scenario can be summarized defining the dataset to give as input to the classifier as:

$$S + T \;\; s.t \begin{cases} \mathcal{R}_{source} & \xrightarrow{zero\ sampling} \;\; S = \emptyset \\ \mathcal{R}_{target} & \xrightarrow{random\ sampling} \;\; T \neq \emptyset \end{cases} \tag{4.3}$$

Figure 4.3 illustrates the scenario described above and clarifies the similarity with a traditional ML algorithm.

4.2.3 Mixing testbed

In the third setting, i.e., the *mixing testbed*, we assume a fixed $S \subseteq \mathcal{R}_{source}$, whereas the set $T \subseteq \mathcal{R}_{target}$ has variable size and is generated via random sampling.

There exist the case $T = \emptyset$, then, only samples from the source domain are leveraged during the training phase.

Note that the sampling from the training set $\mathcal{R}_{source}$ is performed once and then will be fixed for all the future sampling of the same case.

Mixing testbed explores the properties of the cases which combine samples both of the training set source $\mathcal{R}_{source}$ and target $\mathcal{R}_{target}$.

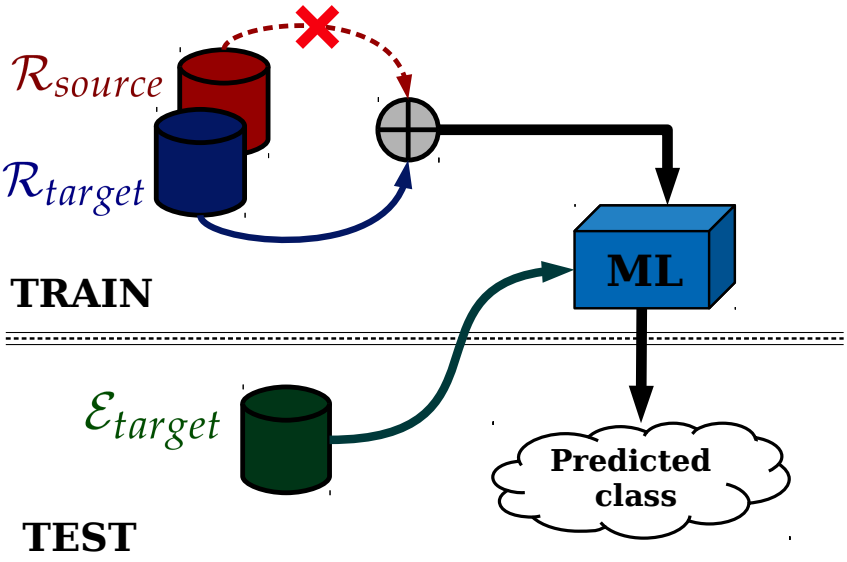


FIGURE 4.3: Only target testbed - Zero samples extracted from the source domain training set and variable number of samples extracted from the target domain training set

In these cases our classifier will receive a larger number of samples, and our analysis focus, in particular, on the ability of the classifier to deal with samples from both domains.

As it will be described in the following setting, mixing testbed is the basic architecture on which the next two testbeds evolve, integrating their proper feature space transformations.

Resulting dataset of the mixing testbed is given by:

$$S + T \;\; s.t \begin{cases} \mathcal{R}_{source} \xrightarrow{random\ sampling} S \neq \emptyset \\ \mathcal{R}_{target} \xrightarrow{random\ sampling} T \neq \emptyset \end{cases} \tag{4.4}$$

Figure 4.4 illustrates the scenario resulting from the above described *data sampling* approach.

4.2.4 Feature Augmentation testbed

The fourth setting, i.e., *Feature Augmentation testbed* is the first setting in which we apply a TL approach over the dataset resulting from the *data sampling* phase.

The construction of sets S and T is analogue to the mixing testbed, so both S and T are *not empty* ($S, T \neq \emptyset$), but once the sampling process is completed, data undergo the feature space transformation procedure in order to perform the Feature Augmentation approach described in Section 3.3.1.

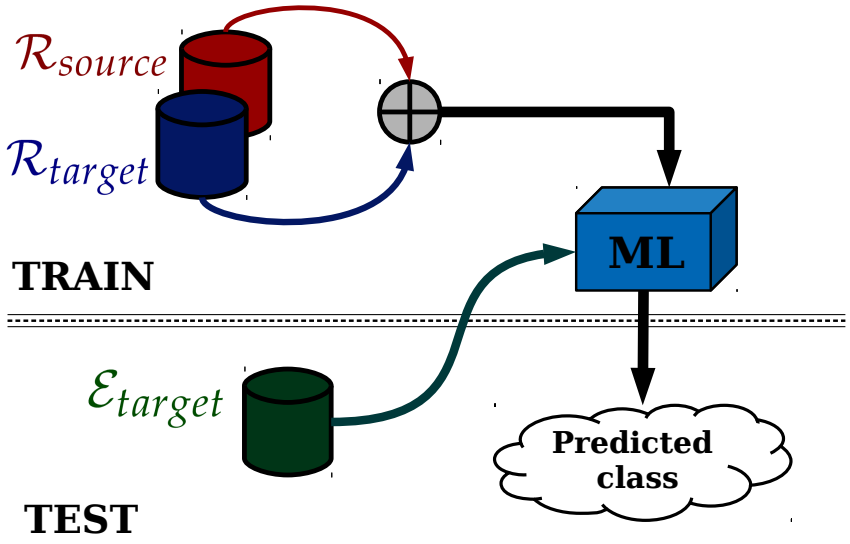


FIGURE 4.4: Mixing testbed - Fixed n° of samples extracted from the source domain training dataset and variable n° of samples, extracted from the target domain training dataset

Figure 4.5 shows, also for this case, a general view of the implemented testbed, including the additional green FA box, which regards the feature space transformation phase that is described Section 3.3.1, proper of each experiment.

4.2.5 CORAL testbed

The fifth setting is the *CORAL testbed*; we consider exactly the same sampling criteria of the Feature Augmentation testbed and so also of the mixing testbed, but we exploit the whole set $\mathcal{R}_{target}$, considering only the features vector and not the label, in the computation of the correlation, to estimate the transformation function to be applied to the samples in S.

Similarly to the previous case, we assume $S \subseteq \mathcal{R}_{source}$, whereas the set $T \subseteq \mathcal{R}_{target}$ has variable size and is generated via random sampling.

The transformation function is used to align the second order statistic of the feature space, i.e., the covariance, according to the process explained in Section 3.3.2.

Transformation step keeps place after applying a *normalization* step to the features of the datasets, i.e., the considered training sets are normalized according to a standard normal distribution with *zero mean* and *unit variance*.

Figure 4.6 shows the characteristics of this setting.

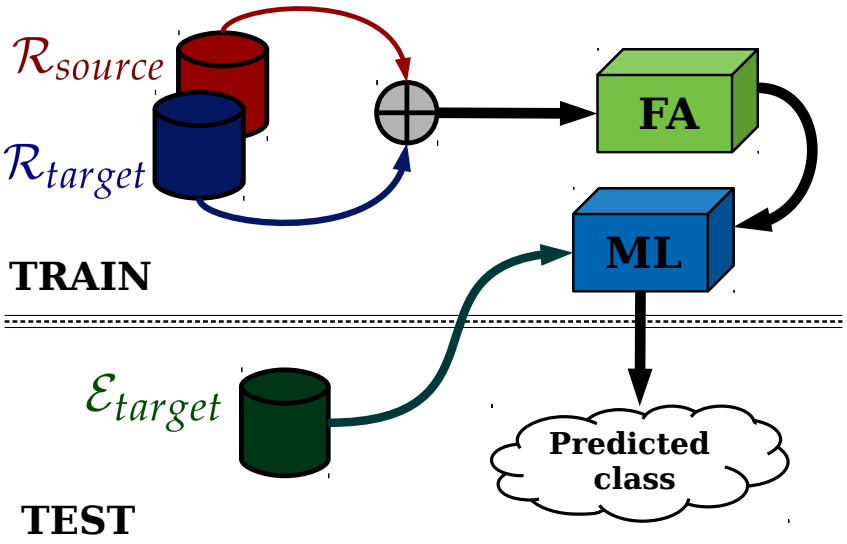


FIGURE 4.5: Feature Augmentation testbed - Fixed n° of samples extracted from the source domain training dataset and variable n° of samples extracted from the target domain training dataset

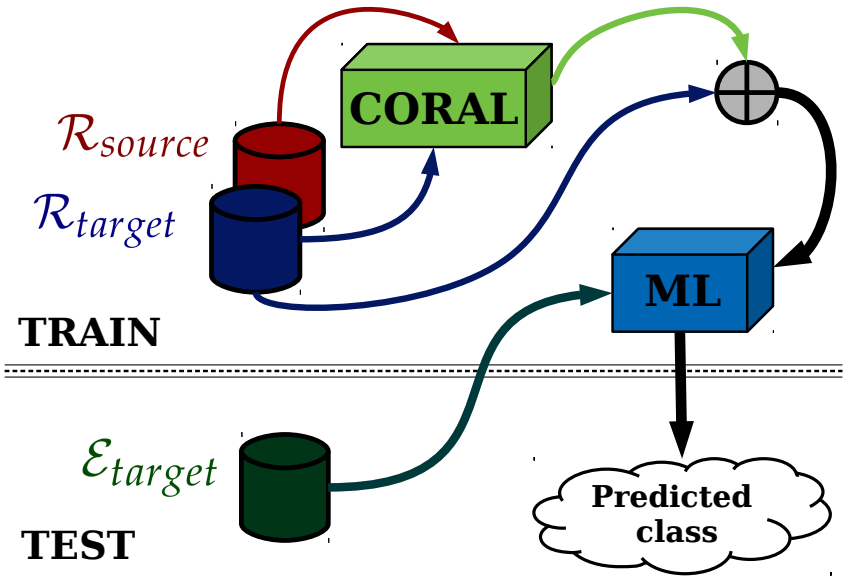


FIGURE 4.6: CORAL testbed - Fixed n° of samples extracted from the source domain training dataset, fixed n° of samples extracted from the target domain training dataset to estimate the transformation function and variable n° of samples extracted from the target domain training dataset

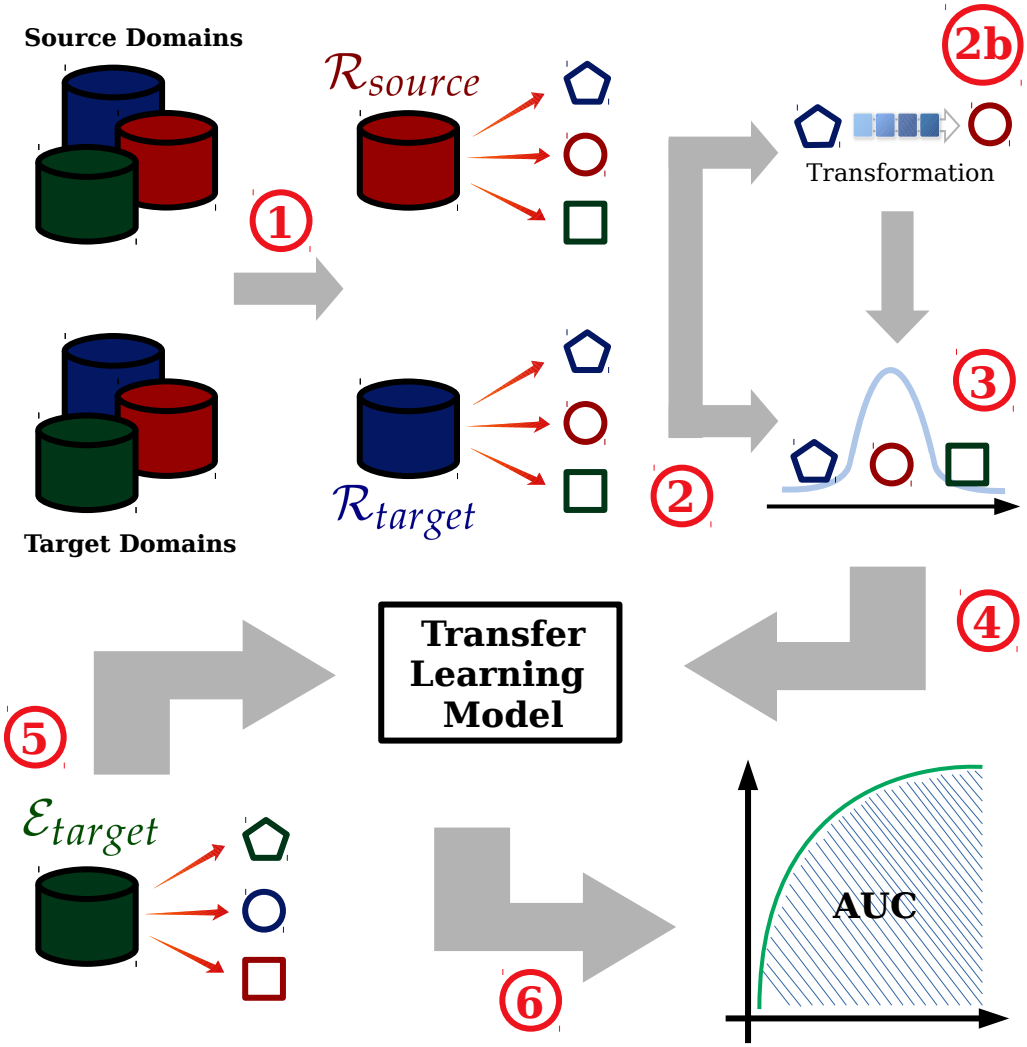


FIGURE 4.7: The experiment phases - 1) Data collection, 2) Data sampling, 2b) (optional) Feature space transformation 3) Data pre-processing, 4) Learning algorithm execution, 5) Learning algorithm test, 6) Performance evaluation

4.3 Design of the experiments

The experiments that have been developed in this work follow a simple scheme: *data collection*, *data sampling*, optional *feature space transformation*, (only for the execution of Feature augmentation and CORAL settings), *data pre-processing*, *transfer learning algorithm execution*, *transfer learning algorithm test* and *performance evaluation*. All of them are detailed in the following subsections, highlighting the key aspects of our implementation.

Figure 4.7 illustrates the fundamental steps of an experiment. We refer to an experiment as the programmed mechanism in which the already described settings are established and execute in turn.

4.3.1 Source domain, Target domain and $\mathcal{R}_{source}$, $\mathcal{R}_{target}$ selection

In this work all the experiments are conducted considering always two domains: a *source* domain and a *target* domain. The selection of source and target domain is identified by one combination among the many datasets generated according to the rules described in Section 3.4 and exploiting the *E-tool*.

Indeed, with E-tool we have generated a collection of training source datasets $\mathcal{C}_{source}$, from which we extract a training source dataset $\mathcal{R}_{source}$ and a collection of training target datasets $\mathcal{C}_{target}$, from which we extract a training target dataset $\mathcal{R}_{target}$.

Each collection of training datasets is composed by unique elements, related to the characteristics of the network topology used to generate them; this ensures that all the source and target training datasets are different:

$$\mathcal{C}_{source} = \{\mathcal{R}_{source1}, \mathcal{R}_{source2}, \ldots, \mathcal{R}_{sourceN}\}, N \in \mathbb{N} \tag{4.5}$$

$$\mathcal{C}_{target} = \{\mathcal{R}_{target1}, \mathcal{R}_{target2}, \ldots, \mathcal{R}_{targetN}\}, N \in \mathbb{N} \tag{4.6}$$

where N is a finite number.

Once the couple $\mathcal{R}_{source}$, and $\mathcal{R}_{target}$ has been selected data can be forwarded to the next phase of the experiment: the *Data sampling*.

4.3.2 Data sampling

During this phase the data inherited from the previous step have to be sampled in order to establish the particular setting explained in Section 4.2.

The *sampling function* $\mathring{f}_s(x)$ is responsible for returning a finite number of *unique* elements x from the original dataset and consists in a *random sampling without replacement* method[1].

In Python, the programming language used to perform experiments, the sampling function $\mathring{f}_s(x)$ that we decided to apply to our datasets is `sample()`.

Function `sample()` invokes the basic function `random()`, which exploits pseudo random numbers generator (RNG) according to *uniform distribution* in the semi-open range $[0.0, 1.0)$.

Marsenne Twister is a complete deterministic core generator and it is widely used as RNG; it produces 53-bit precision floats with a period of $2^{19937} - 1$ and is well suited for our purpose[2].

[1] Sample function
[2] Module random

In our experiments, the *sampling function* $\mathring{f}_s(x)$ is applied twice; once for the *source* domain D_s and once for the *target* domain D_t, returning the following sets of sampled:

$$\mathring{f}_s(x) : \begin{cases} \mathcal{R}_{source} & \longrightarrow \quad S \\ \mathcal{R}_{target} & \longrightarrow \quad T \end{cases} \tag{4.7}$$

The new datasets S and T are then concatenated and ready to be either transformed and pre-processed if the considered settings are Feature Augmentation and CORAL or only pre-processed if only source setting, only target setting and mixing setting have been initialized in the experiment.

4.3.3 Feature space transformation

Feature space transformation is a crucial phase of this thesis, because the assessment of DA techniques performance passes through well designed transformation procedures. It is an optional phase, actually it coincides with the number 2b in Figure 4.7, which is performed by *Feature Augmentation* setting and *CORAL* setting.

If the setting in building is the Feature Augmentation the S dataset and T dataset are processed and their *feature space* will be reconfigured according to the transformation described in Section 3.3.1 in order to obtain a transformed source training set $\tilde{S}$ and a transformed target training set $\tilde{T}$:

$$\begin{cases} S & \xrightarrow{FA} \quad \tilde{S} \\ T & \xrightarrow{FA} \quad \tilde{T} \end{cases} \tag{4.8}$$

Conversely for the CORAL setting, we have to process the same S training source dataset and the same T training target dataset, applying on them the transformation illustrated in Section 3.3.2 in order to obtain the transformed training set of source domain $\tilde{S}$ and the transformed training set of target domain$\tilde{T}$:

$$\begin{cases} S & \xrightarrow{CORAL} \quad \tilde{S} \\ T & \xrightarrow{CORAL} \quad \tilde{T} \end{cases} \tag{4.9}$$

The resulting transformed training source dataset $\tilde{S}$ and transformed training target dataset $\tilde{T}$ are now ready to pass through the following phase of our experiments: *Data pre-processing*.

4.3.4 Data pre-processing

Data sampled and transformed are now ready to pass through the last phase before the training: the *pre-processing step*. This is a common requirement for many ML algorithms, that improves a lot the performance achieved by an estimator.

Unknown distributions that present too dispersed data in the space have points with high variance, e.g., it can happen that a feature f_x, having a variance with an order of magnitude larger than other, can affect the ability of the classifier to learn from others features, because f_x dominates over the others.

In order to avoid this kind of problem and also to facilitate the learning phase of our classifier, in each experiment we perform the pre processing phase of the data, calling the `StandardScaler` class of the *scikit-learn* open source library for Python programming [3].

This class allow us to standardize the collection of features vectors (dataset) through the removal of the mean and the scaling of the variance to one.

In order to implement this procedure, we have to calculate the standard score z for each sample (feature vector) of the dataset:

$$z = \frac{x - \mu}{\sigma} \tag{4.10}$$

where μ is the *mean* value of the training samples, i.e., the feature vector, while σ is the *variance*. Applying this normalization, we center our data to the zero mean value and scale to unit variance the datasets.

As said, the feature scaling results useful for many algorithms, such as K-Nearest Neighbors, Support Vector Machine and Logistic Regression, which require features to be normalized; anyway there exist different feature scaling approaches, e.g., Z-score normalization, which is the one that we use in our experiments, scaling features that lie in a particular range (`MinMaxScaler`), *K-bins discretization*, feature binarization, and so on.

From the practical point of view, since the most machine learning approaches are based on the *euclidean distance* between data points of the distribution used to train the model, the *pre processing phase* reduces the range of the features space distribution of our datasets and circumscribe all the value in a limited interval, according to the type of standardization adopted.

At this point, pre-processed data are ready to be used as input for our classifier, which can perform the training phase under the best conditions.

In the following subsection it will be shown how the training is managed and which kind of classifier is used.

4.3.5 Transfer learning algorithm execution

We are now dealing directly with learning models, in particular, in TL algorithm execution: we provide pre-processed data as input to a classifier to be leveraged during the training phase.

[3] StandardScaler

The training phase coincides with a supervised learning algorithm that configures the internal structure of a model, evaluating samples and relative labels. The aim is to acquire as much knowledge as possible in order to cope with a classification problem.

In this phase of our experiments the training phase consists basically of two parts:

1. definition of the classifier;
2. fitting of the model.

In this thesis, three types of classifiers have been adopted: Random Forest[4] (RF), Logistic Regression[5] (LR) and Support Vector Machine[6] (SVM). The definition of these three classifiers is made exploiting the relative modules of the *scikit-learn* library.

The fitting of the model is realized through the `fit` method, also provided by the *scikit-learn*. It is the crucial instruction in which the model can compare the samples belonging to the transformed source training set $\tilde{S}$, and transformed target training set $\tilde{T}$, with the corresponding label. It corresponds to the core phase of the training and it is the most expensive time procedure of our simulations.

For each experiment, the training phase is executed three times in every setting configuration: the first time to perform the training of a RF classifier, the second time to train a Logistic Regression classifier and the third time to realize the training of a SVM classifier (all algorithms are described in Section 3.1.3).

4.3.6 Learning algorithm test

In our work the testing phase is performed collecting for each source training set $\mathcal{R}_{source}$ and target training set $\mathcal{R}_{target}$ the corresponding target testing set $\mathcal{E}_{target}$.

This phase is mainly characterized by one task:

1. Computing the probability of an instance to belong to one class, rather than to the other one.

The $\mathcal{E}_{target}$ is used to verify the capability of the trained model to classify unlabeled target testing set samples.

In *only target* setting and *mixing* setting, it is possible to identify two limit cases regarding the data sampling, i.e., no samples of the training set source $\mathcal{R}_{source}$ plus the maximum number of samples of training set target $\mathcal{R}_{target}$, and the maximum number of samples of training set source $\mathcal{R}_{source}$ plus no samples of the training set target $\mathcal{R}_{target}$. These two cases identify the *best case* and the *worst case* of the *baseline*, that we use as a fixed reference to compare the achieved performance in the five settings of our learning algorithms.

[4] Random Forest
[5] Logistic Regression
[6] Support Vector Machines

In the following subsection we describe the last phase of the experiment: *performance evaluation*; it regards mainly the metrics adopted to evaluate the output results of the classifier in terms of performance and understand its goodness.

4.3.7 Performance evaluation

The last subsection is characterized by the description of the metrics involved in the quality measurements of our binary classifier, i.e., the *Accuracy* and the *Area Under the Curve (AUC)*[1].

Accuracy

Given the testing set target $\mathcal{E}_{target}$, the accuracy is defined as the ratio of the number of $\mathcal{E}_{target}$ samples that are correctly classified, to the total number of samples of $\mathcal{E}_{target}$. Although it is very intuitive and easily understandable, accuracy suffers of some drawbacks.

The first one is that accuracy is most influenced by the instance number of the two classes present in the test set, i.e., if almost the totality of samples belongs to one class C, the level of accuracy reached by a classifier which always returns the class C will be very high; but this does not provide useful information, nevertheless this measure can be considered as a good metric to evaluate the quality of the classifier.

Another one depends from the value of threshold used to binarize the output probability of the classifier. Indeed, the choice of the threshold affects a lot the value of the accuracy. For example, a classifier that has a constant behavior in the sample assignment of the two classes will lead to low accuracy values if the threshold is set far from the classification scores of the classifier, while lead to high values of accuracy if the threshold is set near to the constant score values.

A third drawback is the difficulty for accuracy to highlight the quality of a classifier that operates in a ambiguous scenario. Consider a test set in which samples are divided into three groups: 25% of true samples, 25% of false samples and a 50% of samples both true and false but very difficult to distinguish among them.

If a classifier C_1 scores 1 for instances of the first group, 0 for instances of the second group and values close to 0.5 for the third one, it is preferable with respect to a classifier C_2 that instead scores 1 and 0 as the previous one, while values near to 0 or 1 for the third group. Both classifiers reach a 75% of accuracy but the second one C_2 manifests a wrong capability to discern instances inside the third group, which are very similar. C_1, instead, exhibits better ability especially for the third group by assigning to them value near to 0.5 and so with better decision quality.

AUC - Area Under the Curve

The other metric, i.e., Area Under the Curve (AUC), tries to solve the above mentioned issues related to the accuracy and it is extremely used in Machine Learning literature. Also in this case, is given a testing set target $\mathcal{E}_{target}$, it is possible to fix an arbitrarly threshold th and divide $\mathcal{E}_{target}$ into four subsets:

1. True Positive (TP) set: the ensemble of samples which are correctly classified because positive by nature and positive by classification;
2. True Negative (TN) set: the ensemble of samples which are correctly classified because negative by nature and negative by classification;
3. False Positive (FP) set: the ensemble of samples which are not correctly classified because negative by nature and positive by classification;
4. False Negative (FN) set: the ensemble of samples which are not correctly classified because positive by nature and negative by classification.

Characterizing the groups helps to inherit other parameters directly connected to the numerosity of *positive* and *negative* samples in the testing set target $\mathcal{E}_{target}$. Indeed, the $TP + FN$ corresponds to the total number of positive (by nature) samples, while $TN + FP$ corresponds to the total number of negative (by nature) samples.

It follows the definition both of *true positive rate (TPR)* and *false positive rate (FPR)*:

$$TPR = \frac{TP}{TP + FN} \tag{4.11}$$

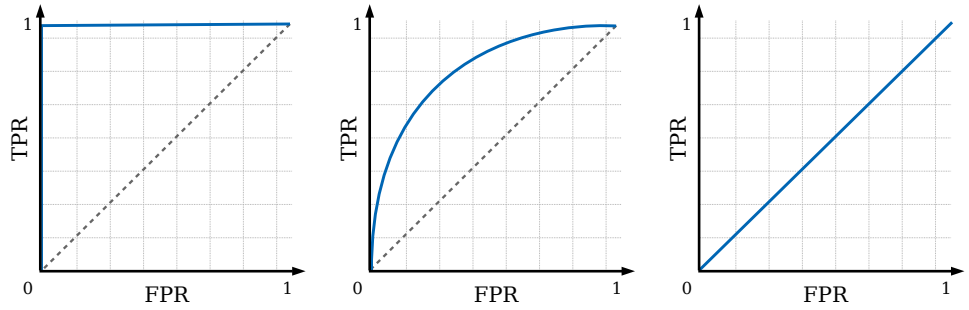
is the fraction between the True Positive (TP) set and the total number of positive (by nature) samples.

$$FPR = \frac{FP}{TN + FP} \tag{4.12}$$

is the fraction between the False Positive (FP) set and the total number of negative (by nature) samples. The domain in which TPR and FPR is comprised between 0 and 1;

Receiver Operating Characteristic (ROC) curve is represented in Cartesian axis, where the x-axis corresponds to the FPR, while the y-axis corresponds to the TPR, both for different values of threshold th. If the threshold th is maximum, i.e., equal to 1, so all samples of $\mathcal{E}_{target}$ are classified as negative, therefore FPR and TPR are very close to zero; vice-versa if threshold th is minimum, i.e., equal to 0, so all samples of $\mathcal{E}_{target}$ are classified as positive, therefore FPR and TPR are very close to one;

By increasing the threshold value th the number of positive classified samples reduces (and number of TP decreases), while the number of negative classified samples augments (and number of TN increases); both TPR and FPR decrease.



(A) AUC = 1 (B) $0.5 < AUC < 1$ (C) AUC = 0.5

FIGURE 4.8: ROC curve - Ideal, Real and Worst cases

By decreasing the threshold value *th* the number of positive classified samples augments (and number of TP increases), while the number of negative classified samples decrements (and number of TN decreases); both TPR and FPR increase.

The ROC curve connects always the extremes (0,0) and (1,1), in the middle we can observe three different scenarios:

- Figure 4.8a - ROC curve connecting (0,0), (0,1), (1,1) $\xrightarrow{Ideal}$ AUC = 1;
- Figure 4.8b - ROC curve in the middle $\xrightarrow{Real}$ $0.5 < AUC < 1$;
- Figure 4.8c - ROC curve connecting (0,0), (1,1) $\xrightarrow{Worst}$ AUC = 0.5.

The AUC does not depend on the choice of the threshold *th*, it is robust and it is recommended to be used to evaluate binary classifiers[7].

Chapter 5

Numerical Assessment

This chapter provides a full overview of all the experiments done in this work and exhibits the numerical assessment for each of them.

Our analysis will move from the generation of the datasets, and their technical description, i.e., the network topologies to which are related, the scaling factor adopted to resize them, the total samples size, the number of samples per class.

After describing the complete pool of data that have been used in the experiments, we consider different combinations of source and target domain datasets and apply the TL learning models described in Sections 4.2.4 and 4.2.5.

The performance analysis considers the two metrics described in Section 4.3.7, which clarify the different behavior of the classifiers with respect to the different considered domains.

In the first part of the chapter, results obtained wuth the Random Forest classifier and the simple swapping of source domain and target domain are obtained and compared to those presented in [16], while in the second part, the other two classifiers, i.e., SVM and Linear regression, are also adopted and their behavior is evaluated under all the settings described in Chapter 4.

As last step, we investigate how the performance values are related to the distance that exists between the lightpath length distributions of the source dataset and target dataset.

5.1 Datasets description

For the generation of the datasets[1] we refer to a transmission system in which optical channels have a slice width of 12.5 GHz [18] and elastic tranceivers operate at 28 Gbaud with a bandwidth of three slices, i.e., 37.5 GHz. Modulation formats are chosen among dual polarization (DP)-BPSK, QPSK and n-QAM, with n = 8, 16, 32, 64. For traffic demands that exceed the transceiver bandwidth, multiple adjacent

[1]The generation of the dataset has been done, making use of the cluster of the High Performance Computing (HPC)[9] center of the Politecnico di Torino, running Matlab scripts.

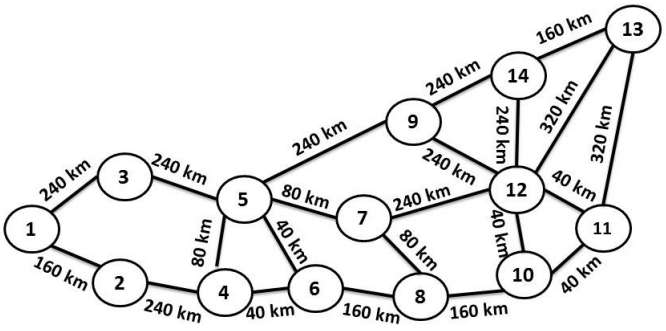


FIGURE 5.1: Japan Network Topology

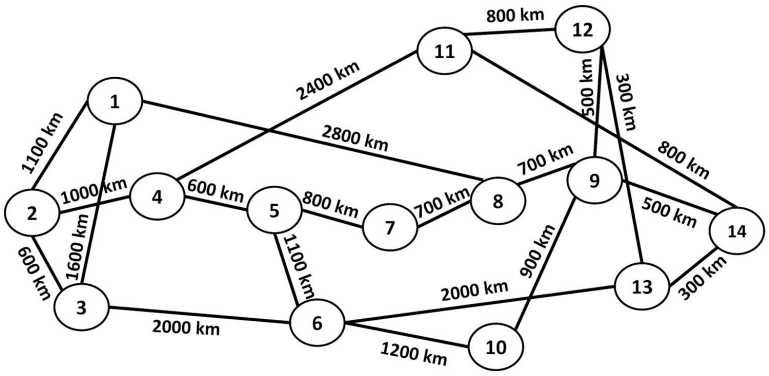


FIGURE 5.2: NSF Network Topology

transceivers are used. We consider standard single mode fiber (SMF) transparent links, where equally spaced optical amplifiers over the links restore the signal power every 100 Km. Other parameters setting are indicated in Section 3.4.

We consider the Japan and NSF networks shown in Figure 5.1 and 5.2, respectively, where the reported link lengths, refer to the original topology (not re-scaled). In order to get additional datasets, the two original topologies have been multiplied by a scaling factor (SF), that either magnifies or shrinks the link lengths.

Table 5.1 lists the 16 datasets corresponding to the training set $\mathcal{R}$ and testing set $\mathcal{E}$ used to perform our experiments, indicating the considered topology, scaling factor value, total size and percentage of samples which exhibit a BER above the fixed threshold value γ. The scaling factor (SF) indicates the multiplicative factor applied to each link of the topology to obtain a re-scaled version.

The train sets $\mathcal{R}$ are generated according to the associated topology structure, we produce 10000 instances by randomly selecting a source-destination node pairs, a modulation format and a traffic demand, which is uniformly extracted from a range of $[50 - 500]$ Gbps, with 50 Gbps granularity. Finally, the BER is evaluated with the E-tool described in Section 3.4. During the generation of Japan*, the parameter setting is slightly different, indeed the fiber attenuation per km changes from 0.2 to 0.25, and the Noise Figure amplifier from 5 dB becomes 6 dB.

TABLE 5.1: Dataset Description - Train Set $\mathcal{R}$ and Test Set $\mathcal{E}$

$\mathcal{R}$ / $\mathcal{E}$	**Topology**	**SF***	**Total size**	**Above γ - (%)**
$\mathcal{R}$	Japan	0.5	10000	25.04
$\mathcal{R}$	Japan	1	90000	40.97
$\mathcal{R}$	Japan	2	10000	63.92
$\mathcal{R}$	Japan*	1	10000	59.93
$\mathcal{R}$	NSF	0.2	10000	62.25
$\mathcal{R}$	NSF	0.5	10000	78.97
$\mathcal{R}$	NSF	1	90000	89.91
$\mathcal{R}$	NSF	2	10000	96.69
$\mathcal{E}$	Japan	0.5	90000	55
$\mathcal{E}$	Japan	1	90000	$42.0\bar{8}$
$\mathcal{E}$	Japan	2	90000	$62.\bar{1}$
$\mathcal{E}$	Japan*	1	90000	$59.68\bar{6}$
$\mathcal{E}$	NSF	0.2	90000	67
$\mathcal{E}$	NSF	0.5	90000	$82.33\bar{1}$
$\mathcal{E}$	NSF	1	90000	$90.53\bar{8}$
$\mathcal{E}$	NSF	2	90000	$97.47\bar{3}$

*Scaling Factor;
*$\gamma = 4 \cdot 10^{-3}$.

The test sets $\mathcal{E}$ follow the same rules of the train sets generation, the difference is that in this case the total sample size is 90000.

The BER threshold value γ selected is $4 \cdot 10^{-3}$ for both train sets $\mathcal{R}$ and test sets $\mathcal{E}$.

For each train set $\mathcal{R}$, we compute the number of samples which are above or below the BER threshold γ, this is done through the use of a logarithmic scale which improves the readability of these data. We want to explore how the number of *True* samples, i.e., below γ, and the number of *False* samples, i.e., above threshold γ, can affect the performance of our classification problem. For this reason, Appendix A reports the logarithmic histograms for each train set $\mathcal{R}$, that highlight the number of samples for different ranges of BER values.

The total number of sample size below the threshold γ keep decreasing when augmenting the scaling factor (SF), so the more a network become larger, the more the number of samples below the threshold γ decreases. Vice-versa, the total number of sample size above the threshold γ keep increasing when augmenting the scaling factor (SF).

This trend is confirmed in both the network topologies reported in Appendix A: the maximum value of samples above the threshold belong to the Japan topology, re-scaled according to $SF = 0.5$, as illustrated in Figure A.1a. It has 7496 true values, with respect to the 2504 samples, which instead are False.

The minimum value of True samples is reached by the NSF topology, re-scaled according to $SF = 2$, i.e., all link lengths are doubled. In Figure A.2d is shown

that, for this topology, the number of True samples is 331.

Generally, the Japan topology exhibits more True values with respect to the NSF topology: this is due to the smaller size of the Japan network in comparison to the NSF network.

For example, the smallest NSF topology, i.e., the NSF re-scaled with $SF = 0.2$, has similar size of the largest Japan topology, i.e., Japan re-scaled with $SF = 2$, and for the first, the number of True samples is 3775 (Figure A.2a), while for the second, the number of True samples is 3608 (Figure A.1c); the two values are very similar due to the similar size of the two topologies.

For each train set $\mathcal{R}$ we also evaluated the probability distribution function of the lightpath lengths associated to each source-destination node pair.

Figures B.1 and B.2 reported in Appendix B provide a representation of the probability distribution function of lightpath lengths for each combination of the two topologies. As said before, the Japan topology is in general much smaller than NSF topology, so for networks topologies of very different sizes, the shared support between the two is minimal. In this initial stage, we are interested in the evaluation of the similarity of two topologies in terms of shared area between distributions, so the more the shared area is large, the more the two distributions are similar.

Observing the graphs in Appendix B, the distribution of each topology scales according to the SF, but does not change its shape. This is due to the fact that, we design our generation process in order to create re-scaled datasets which recall to the original ones by the selection of the same routes among the nodes.

Borderline cases for different network topologies correspond to Figure B.1d, in which the area shared by the two distributions is very small, and on the contrary, to Figure B.2c, in which instead, the support of the two distribution is almost the same and the shared area by the distribution is big.

The 12 combinations of train set $\mathcal{R}$, resulting by the 3 versions of re-scaled Japan topologies and the 4 versions of re-scaled NSF topologies are listed in the Table 5.2, where the minimum lightpath length for both topologies, the maximum lightpath length for both topologies and their corresponding average values are indicated. The probability distribution function (pdf) of the lightpaths is obtained considering histograms with 300 Km binwidth.

5.2 A first approach with Machine Learning

We initially consider the same scenario proposed in [16] and replicate the results therein reported by training of a Random Forest (RF) classifier with samples belonging to a given topology and performing the test with samples of the same topology.

TABLE 5.2: Lightpaths statistics - Minimum, Maximum and Mean

$\mathcal{R}_1 - SF_1$ $\mathcal{R}_2 - SF_2$	**Min. length (Km)**	**Max. length (Km)**	**Mean value (Km)**
Japan - 0.5 NSF - 0.2	20 80.0	640 2628.7854	254.21 995.059
Japan - 0.5 NSF - 0.5	20 150	640 4250	254.21 1799.38
Japan - 0.5 NSF - 1	20 300	640 8500	254.21 3598.76
Japan - 0.5 NSF - 2	20 600	640 17000	254.21 7197.52
Japan - 1 NSF - 0.2	40 80.0	1280 2628.7854	508.42 995.059
Japan - 1 NSF - 0.5	40 150	1280 4250	508.42 1799.38
Japan - 1 NSF - 1	40 300	1280 8500	508.42 3598.76
Japan - 1 NSF - 2	40 600	1280 17000	508.42 7197.52
Japan - 2 NSF - 0.2	80 80.0	2560 2628.7854	1016.84 995.059
Japan - 2 NSF - 0.5	80 150	2560 4250	1016.84 1799.38
Japan - 2 NSF - 1	80 300	2560 8500	1016.84 3598.76
Japan - 2 NSF - 2	80 600	2560 17000	1016.84 7197.52

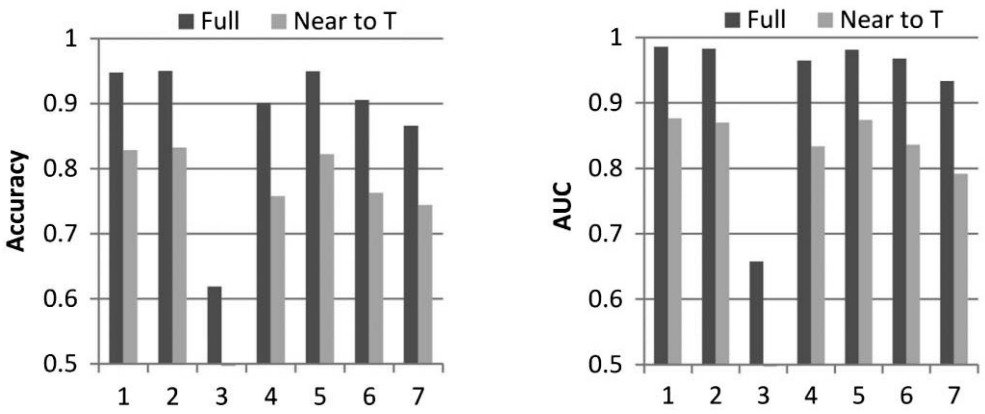
In [16] this kind of approach has been then extended in order to perform an analysis on the feature relevance. In particular, the selected subsets of features are represented in Table 5.3 and are used in turn, to train the classifier with 1000 training samples. The considered features are those enumerated in Section 4.1.

Figure 5.3 and 5.4 report the histograms that show the performance achieved by the classifier in terms of *Accuracy* and *AUC* with the Japan and NSF topologies assuming SF=1, for the scenario test with the complete $\mathcal{E}$ and for the scenario test with the *near to threshold* samples of the $\mathcal{E}$. The *near to threshold* identifies the dataset $\mathcal{E}$ which has been filtered keeping the samples that are close to the threshold value γ. In particular, filtering is performed selecting only the values in between the interval $[4 \cdot 10^{-4}, 4 \cdot 10^{-2}]$ to investigate on the special scenario in which the instances are difficult to classify.

From the results depicted in Figure 5.3 and 5.4, it is possible to note that in Japan topology the difference in terms of both AUC and Accuracy of *near to threshold* samples is not pronounced as in the NSF topology. In addition, it results clear

TABLE 5.3: Feature Subsets

Features	S1	S2	S3	S4	S5	S6	S7
Number of links	✓	✓	✓	✓			
Lightpath length	✓	✓	✓	✓	✓	✓	
Length of longest link	✓	✓	✓	✓			
Traffic volume	✓	✓	✓		✓		✓
Modulation format	✓	✓		✓	✓	✓	✓
Guardband, modulation format and traffic volume of nearest left and right neighbor	✓						



(A) Accuracy, depending on the feature set selection relative to Japan topology

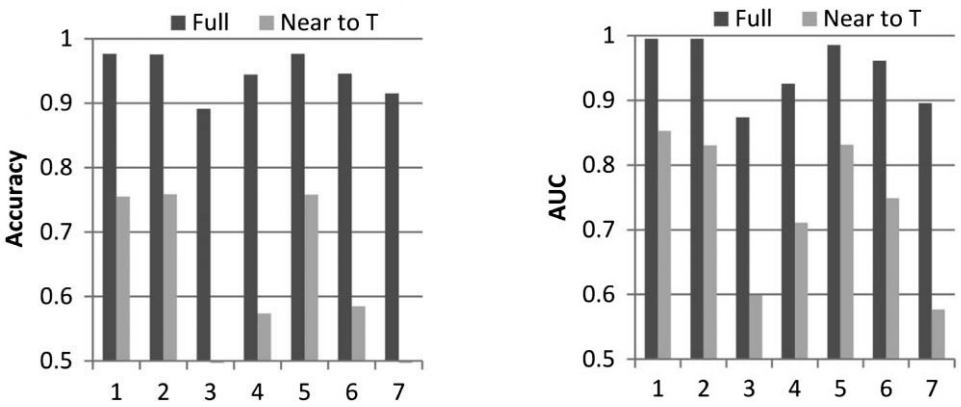
(B) AUC, depending on the feature set selection relative to Japan topology

FIGURE 5.3: Performance evaluation w.r.t feature selection using $\mathcal{R}_{Japan}$ and $\mathcal{E}_{Japan}$[16]

the impact of particular subsets of feature with respect to others; for example the lowest performance are achieved when *modulation format* attribute is not considered, i.e., subset *S3*, while the highest performance are obtained when subsets S1, S2, S5 are considered. The training with the features which are proper of the links, i.e., *lightpath length*, *traffic volume* and *modulation format* (subset *S5*) allows to reach good performance in both topologies and are comparable with the subsets that exploit all the 11 features (S1) and subset *S2*, which adds *number of links* and *length of the longest link* attributes. Readers interested in learning more about these results, are invited to read the complete description given in [**Section C.**, 16]. Based on the described results, we decided to adopt only *S*1 and *S*2 subsets.

5.2.1 Changing the target domain

Maintaining the same Random Forest classifier of [16] for the sake of comparison, which exploits a forest of 25 decision trees, we repeat the same experiments illustrated above, but using the test $\mathcal{E}_{target}$ of a different domain. So, for example, if subset *S1* is



(A) Accuracy, depending on the feature set selection relative to NSF topology

(B) AUC, depending on the feature set selection relative to NSF topology

FIGURE 5.4: Performance evaluation w.r.t feature selection using $\mathcal{R}_{NSF}$ and $\mathcal{E}_{NSF}$ [16]

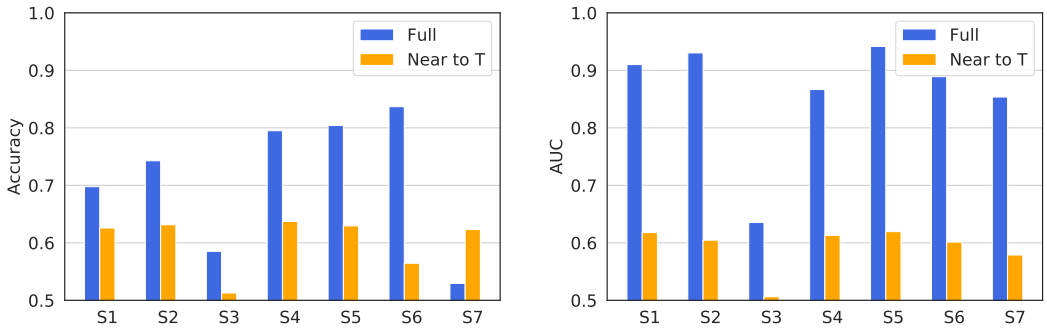
used to train the RF classifier with samples drawn from the Japan topology, then the test is performed using the target test set $\mathcal{E}_{target}$ of the NSF topology.

Intuitively, this is a bad scenario to apply a machine learning algorithm, because a ML model is able to recognize samples very similar to those used for the training.

$\mathcal{R}_{Japan}$ and $\mathcal{E}_{NSF}$

Figure 5.5 shows the results obtained training a RF classifier with $\mathcal{R}_{Japan}$, and testing it with $\mathcal{E}_{NSF}$, following the same procedure of the previous experiments. In comparison to results plotted in Figure 5.3 the performance reached both by Accuracy and AUC consistently decreases, but the AUC exhibits a better trend in the full test condition, confirming that the best scenarios under which perform the train are: S1, S2, S5.

For the near to threshold cases, the performance degradation is even more pronounced both for AUC and Accuracy, and the discrepancy between the full test scenario and near to threshold scenario is much more evident.



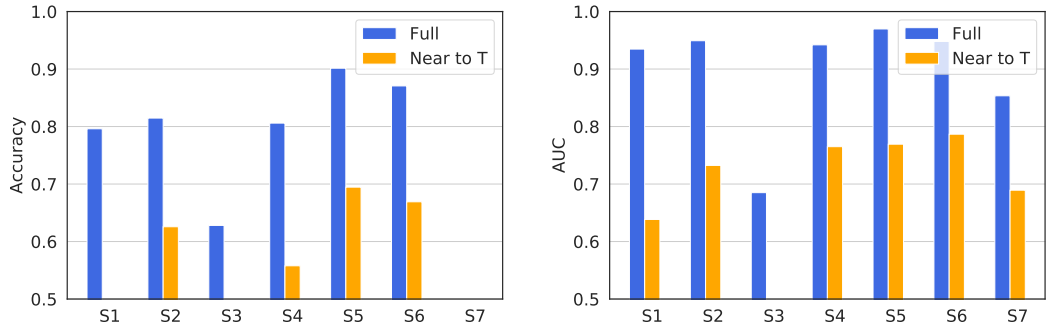
(A) Accuracy, depending on the feature set selection relative to Japan topology

(B) AUC, depending on the feature set selection relative to Japan topology

FIGURE 5.5: Performance evaluation w.r.t feature selection using $\mathcal{R}_{Japan}$ and $\mathcal{E}_{NSF}$

$\mathcal{R}_{NSF}$ and $\mathcal{E}_{Japan}$

Figure 5.6, instead, shows the results of the opposite case in which $\mathcal{R}_{NSF}$ is used as training set, while $\mathcal{E}_{Japan}$ is used as test set. The Accuracy touches low values both for full test scenario and near to threshold and the subset $S5$, seems to better behave with respect to others. The AUC instead, maintains higher values, especially considering the full test scenario and making the training with S1, S2, S5 subsets.



(A) Accuracy, depending on the feature set selection relative to NSF topology

(B) AUC, depending on the feature set selection relative to NSF topology

FIGURE 5.6: Performance evaluation w.r.t feature selection using $\mathcal{R}_{NSF}$ and $\mathcal{E}_{Japan}$

Results reported in Figures 5.5-5.6 allow us to quantify the impact of changing the domain where a trained ML model is applied, thus motivating the need for the adoption of TL methodologies. Table 5.4, reports the difference of AUC and Accuracy values between the testing made on the same network versus the testing made on different network is quantified in percentages.

TABLE 5.4: Accuracy and AUC discrepancy (%) between testing on the same network vs different network

			S1	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**
JJ-JN	Accuracy	Full	22.03	18.61	4.94	7.15	11.89	3.73	35.78
		Near	20.61	21.83	4.29	15.38	22.32	24.33	13.52
	AUC	Full	3.51	2.5	0	5.52	0.90	3.62	5.66
		Near	27.20	29.94	7.44	24.47	28.79	25.99	27.26
NN-NJ	Accuracy	Full	11.02	10.72	0	5.87	1.21	0	43.28
		Near	41.29	22.53	26.12	25.92	14.28	10.29	56.11
	AUC	Full	0.89	0.51	0	0	0	0	5.67
		Near	24.78	15.14	32.66	5.75	11.57	3.16	13.38

JJ=$\mathcal{R}_{Japan}\mathcal{E}_{Japan}$
NN=$\mathcal{R}_{NSF}\mathcal{E}_{NSF}$
JN=$\mathcal{R}_{Japan}\mathcal{E}_{NSF}$
NJ=$\mathcal{R}_{NSF}\mathcal{E}_{Japan}$

5.2.2 Receiver Operating Characteristic curves

Another element of comparison among the results obtained in [16] and ours, consists in the representation of the ROC curves.

Several trials have been conducted modifying the samples size of training sets $\mathcal{R}$, in particular our training datasets are composed by 90000 instances each, and we evaluate the impact of the training set size in five different cases, sampling the $\mathcal{R}$ in a subset $\bar{\mathcal{R}} \mid \bar{\mathcal{R}} \subseteq \mathcal{R}$:

- Train Set $\bar{\mathcal{R}}$ of *10* samples;
- Train Set $\bar{\mathcal{R}}$ of *100* samples;
- Train Set $\bar{\mathcal{R}}$ of *1000* samples;
- Train Set $\bar{\mathcal{R}}$ of *10000* samples;
- Train Set $\bar{\mathcal{R}}$ of *90000* samples.

Intuitively, the increment in the training set size should yield to a better performance, and, consequently, a higher AUCs. ROC curves appeared in [16] are illustrated in Figure 5.7a and 5.7b, and are used for comparison to our scenarios, in which the target domain instead, $\mathcal{E}$ are swapped.

As illustrated in Figure 5.7c and Figure 5.7d, though the AUC values grow with increasing train set size, by simple visual inspection it results clear that changing target domain leads to a consistent reduction of AUC values, w.r.t. Figures 5.7a and 5.7b.

Note that the trend of the brown ROC curves in Figures 5.7c and 5.7d follows that of black ROC curve; this fact indicates that using a train set size 90000 samples does not significantly improve the performance of our classifier with respect to using a train set size of 10000 samples. For this reason, from now on we can consider a training set $\mathcal{R}$ of 10000 samples.

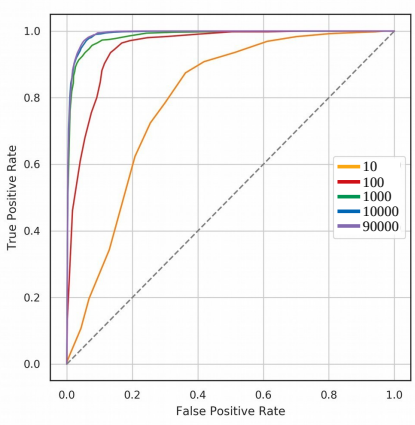
This important aspect of our investigation has been used in order to built our baselines and obtain the results reported in the next subsections.

5.3 Baselines assessment

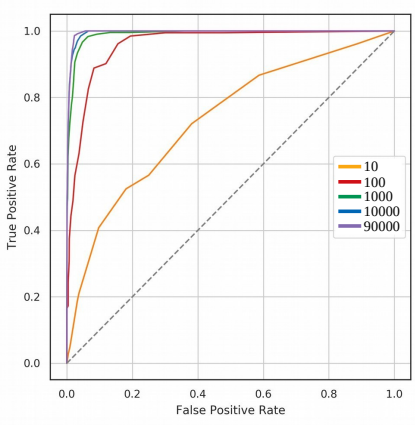
We now report a first evaluation of the performance improvements achieved by the simplest DA technique described in Section 4.2.3,i.e., dataset mixing.

The baseline assessment consists into twelve study cases which are referred to the *only target testbed* and the *mixing testbed* described in the previous Sections 4.2.2 and 4.2.3.

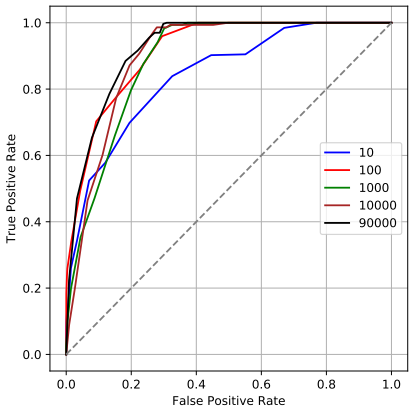
In particular, in the *only target* setting there are no instances from the source domain R_{source}, whereas we consider 10, 50, 100 500, 1000 and 10000 samples from target



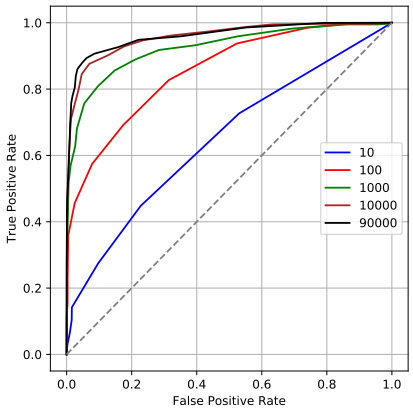
(A) ROC curve for different $\mathcal{R}$ size over Japan topology



(B) ROC curve for different $\mathcal{R}$ size over NSF topology



(C) ROC curve for different $\mathcal{R}$ size over Japan topology and testing with NSF



(D) ROC curve for different $\mathcal{R}$ size over NSF topology and testing with Japan

FIGURE 5.7: ROC curves for the impact of Train Set $\mathcal{R}$ size

domain R_{target}, for a total of six different cases, formally:

$$S + T \;\; s.t \begin{cases} \mathcal{R}_{source} & \xrightarrow{zero\ sampling} \;\; S = \emptyset \\ \mathcal{R}_{target} & \xrightarrow{random\ sampling} \;\; T = i, \;\; i = 10, 50, 100, 500, 1000, 10000 \end{cases} \tag{5.1}$$

Intuitively, performance metric are expected to grow when increasing the number of samples in T, from 10 up to the maximum value.

Last case, i.e., $T = 10000$ samples, corresponds to the best case scenario, in which the classifier can be trained and tested with all data of target domain.

The *mixing* setting is composed, instead, by the last six cases, in which 10000 fixed instances of the source domain $\mathcal{R}_{source}$ are combined with 0, 10, 50, 100, 500 and 1000 samples of target domain $\mathcal{R}_{target}$, formally:

$$S + T \;\; s.t \begin{cases} \mathcal{R}_{source} & \xrightarrow{fixed\ sampling} \;\; S = 10000 \\ \mathcal{R}_{target} & \xrightarrow{random\ sampling} \;\; T = i, \;\; i = 0, 10, 50, 100, 500, 1000 \end{cases} \tag{5.2}$$

Once sampled, the 10000 samples of S remain fixed for all the cases, while the target instances T keep growing time by time from 10 to 10000.

Note that when $i = 0$, the training step is performed ones with samples of the source domain, while testing step is executed with samples belonging to target domain; therefore, it is expected that the classification is more difficult for the model and furthermore the performance should be worse with respect to other cases, where some samples drawn from the target domain are also available for the training phase. For this reason, we identify this particular case study as the worst case scenario.

Results that we are going to show merge the two settings in a single graph, where the achieved values of AUC and Accuracy are compared taking into consideration the following scenarios:

- **Japan-to-NSF**: The considered *source domain* is the Japan network while the considered *target domain* is the NSF network. To perform the training $\mathcal{R}_{Japan}$ is selected as source train set and $\mathcal{R}_{NSF}$ as target train set, while the testing is performed with the complete target test set $\mathcal{E}_{NSF}$ of 90000 instances;
- **NSF-to-Japan**: The considered *source domain* is the NSF network while the considered *target domain* is the Japan network. To perform the training is used $\mathcal{R}_{NSF}$ as source train set, and $\mathcal{R}_{Japan}$ as target train set, while the testing is performed with the complete target test set $\mathcal{E}_{Japan}$ of 90000 instances.

The *only target* setting is identified by the blue bar and each case is labeled with the notation xNSF, in the **Japan-to-NSF** scenario or xJAP, in the **Japan-to-NSF**, where x denotes the number of sampled instances from the $\mathcal{R}_{NSF}$ or $\mathcal{R}_{Japan}$, respectively.

The *mixing* setting is identified by the yellow bar and each case is labeled with the notation $x$all, in both **Japan-to-NSF** and **NSF-to-Japan** scenarios, where x denotes the number of sampled instances from the target $\mathcal{R}_{NSF}$ or target $\mathcal{R}_{Japan}$, respectively.

Best case and worst case are depicted with dashed red and blue lines, respectively.

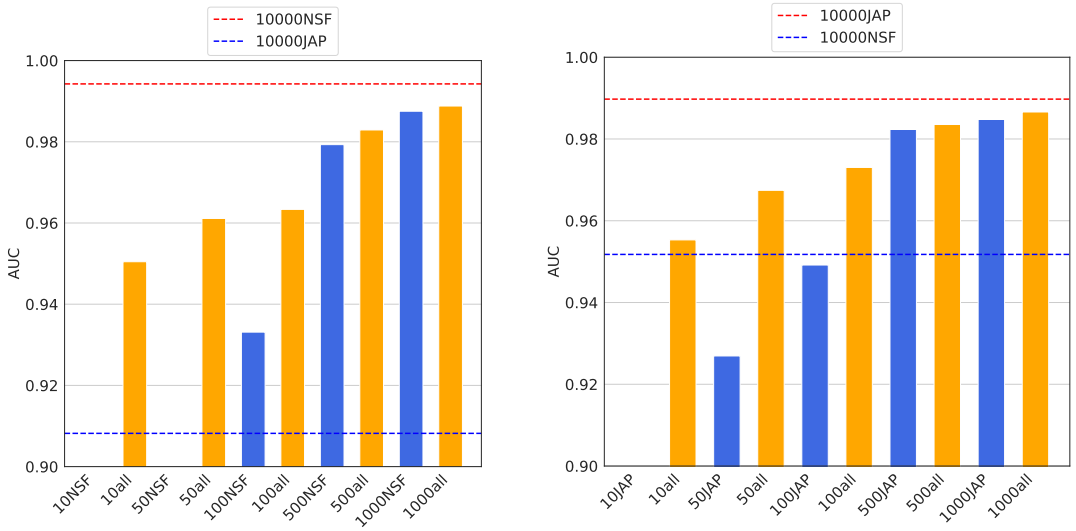
From now on, all the results that will be provided are averaged are averaged base on 20 replicas of the same experiments.

Figure 5.8 illustrates the bars that represents the performance value of AUC achived in the scenario **Japan-to-NSF** and in the scenario **NSF-to-Japan**, respectively. Figure 5.8a, shows a *best case* close to one, while a *worst case* baseline near to 0.9. In between we can observe an increasing trend according to the growth of the samples size.

Results obtained in the mixing setting always outperform those obtained in the only target setting.

For what concerns instead the Figure 5.8b, it is possible to note a higher value of the worst case AUC, and the yellow bars are significantly higher than the blue bars

especially in the cases with few sample of the target training set are considered; conversely, for high number of samples, i.e., 500 and 1000, the behavior of the yellow and blue bars seems to be comparable in both scenarios.



(A) AUC - Japan-to-NSF

(B) AUC - NSF-to-Japan

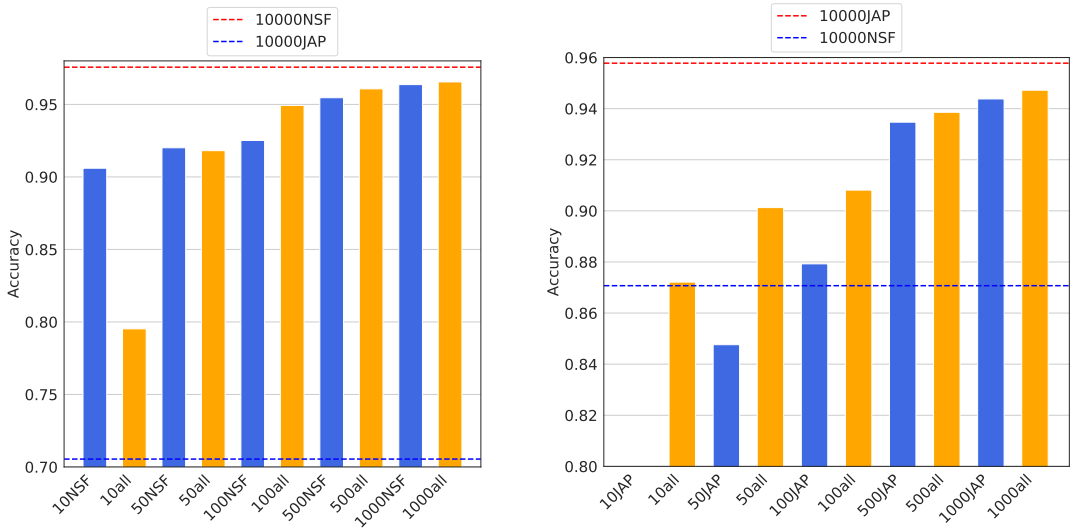
FIGURE 5.8: Baseline Japan-to-NSF / NSF-to-Japan - Bars, Area Under the Curve (AUC) comparison

Figure 5.9 presents analogous results considering the Accuracy metric, for the **Japan-to-NSF** and the **NSF-to-Japan** scenarios. In particular, Figure 5.9a, shows that also the Accuracy presents an increasing trend for both blue and yellow bars. Best case accuracy is in between 0.97 and 0.98. Moreover we can observe that the *mixing* testbed beats the *only target* testbed almost in all the cases, except for the first two ones, i.e., with 10 and 50 samples.

For what concerns instead Figure 5.9b, Accuracy values are generally lower than in the **Japan-to-NSF**. Here, *mixing* testbed always outperforms the blue bars. From the graph of the distribution B.2a, it results evident that the NSF network, given its bigger dimension, contains lightpath lengths that can be established in Japan topology, so the training performed under *mixing* setting, brings benefits to the capacity of the classifier to get better performance.

This intuition is valid in general, because performing the training in a set S and test the model in a subset T can be useful because surely the model has seen the type of data in T, vice-versa, the probability that a model has seen the type of data of S, after the training in T, is related to the grade of similarity or dissimilarity, or the distance, that exists between the two sets.

Last task of baseline analysis regards the study of the distributions of the obtained performance values, both for the AUC and Accuracy. To this aim, we provide boxplots of the AUC and Accuracy values obtained in the scenarios already described.



(A) Accuracy - Japan-to-NSF

(B) Accuracy - NSF-to-Japan

FIGURE 5.9: Baseline Japan-to-NSF / NSF-to-Japan - Bars, Accuracy comparison

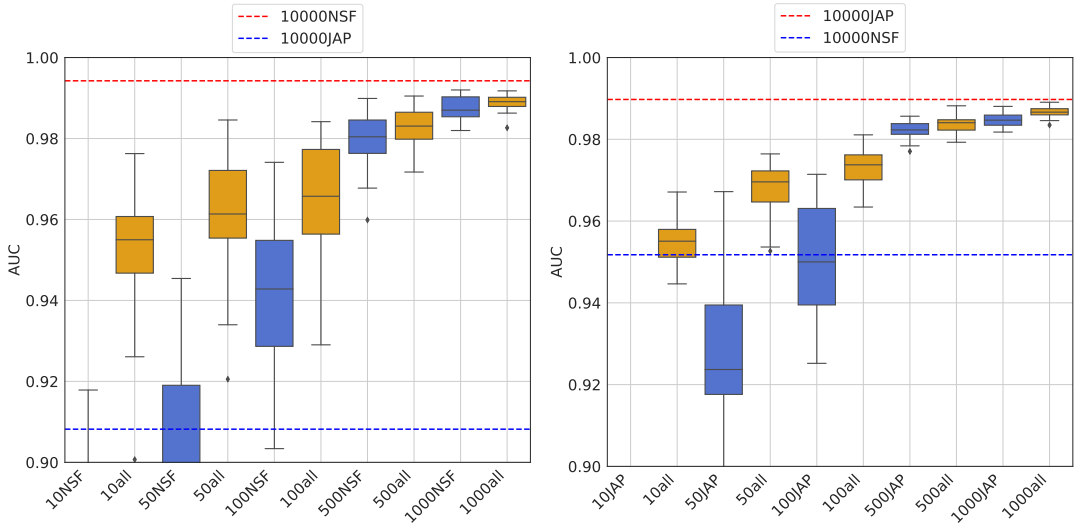
In general, in both the *only target* (blue) and *mixing* (yellow) settings, boxplots become narrower with the increase of the number of samples; this is due to the fact that we run the same experiments 20 times and so, every time, the sampling procedure selects a different target train set $\mathcal{R}_{target}$; as a consequence, the cases in which the number of sampled instances is low, manifest an higher variance in the space. This is true, especially for the *only target* setting, because we perform the training with at most 1000 samples (excluding the best case scenario), while in *mixing* setting, the minimum number of fixed samples is given by the source train set $\mathcal{R}_{source}$.

Figure 5.10 shows the boxplots of AUC for each case of every setting, while Figure 5.11 shows the boxplots of Accuracy: the trend follows that of the bars in Figure 5.8 and 5.9; as said before, we can notice an interesting characteristic, common to both the *only target* and *mixing* settings, i.e., the narrowing of the boxplots with the increase of the number of samples. The *mixing* setting case always outperforms the *only target* setting.

Int he next subsection, we explore the performance of the other two DA techniques (i.e., FA and CORAL) to see if they can outperform the *mixing* approach.

5.4 Domain adaptation assessment

We have seen that with a Random Forest classifier is possible to achieve good levels of AUC and Accuracy, in this section we provide results obtained performing two DA techniques: Feature Augmentation and CORAL. So, we extend the graph reported in the previous subsection including the FA setting and the CORAL setting, in which the feature transformation keeps place. The Feature Augmentation setting is labeled by Taug, where $T \in \{10, 50, 100, 500, 1000\}$ corresponds to the sampled instances



(A) AUC - Japan-to-NSF

(B) AUC - NSF-to-Japan

FIGURE 5.10: Baseline Japan-to-NSF / NSF-to-Japan - Boxplots, AUC comparison

from the target set $\mathcal{R}_{target}$. Actually, we decided to remove out the case in which $T = 1000$ because it behaves approximately as the case $T = 500$.

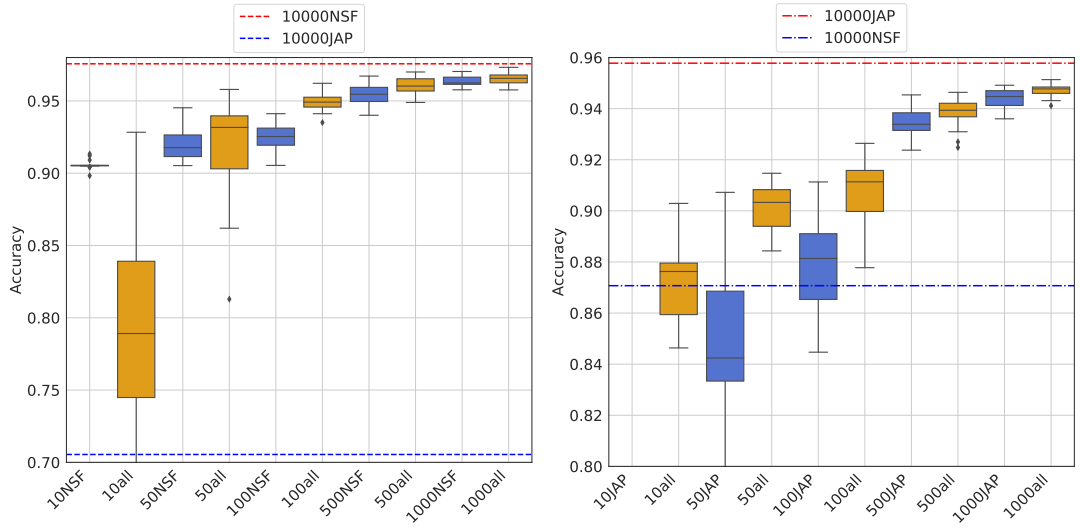
For what concerns the best case scenario, i.e., when considering 10000 samples of target set $\mathcal{R}_{target}$, we adopted a dashed dot line, while for the worst case one, i.e., when are considered only 10000 source set $\mathcal{R}_{source}$ samples, we opted for a simply dotted line.

For CORAL setting we made the assumption that 10000 *unlabeled* instances of $\mathcal{R}_{target}$ are used to re-color source data; note that, for the QoT prediction task, collecting unlabeled samples of the target set $\mathcal{R}_{target}$ is trivial, as we simply need to select route, traffic volume and modulation format of a perspective ligthpath to derive its feature vector, but we do not need to measure its BER.

We refer to this particular case with the notation xCOR, where x corresponds to the sampled instances of the target set $\mathcal{R}_{target}$ which are appended to the 10000 instances of the transformed source set $\mathcal{R}_{source}$.

As done for baseline assessment, we conduct our experiment considering the two scenarios:

- **Japan-to-NSF**: The considered *source domain* is the Japan network while the considered *target domain* is the NSF network;
- **NSF-to-JAP**: The considered *source domain* is the NSF network while the considered *target domain* is the JAP network.



(A) Accuracy - Japan-to-NSF

(B) Accuracy - NSF-to-Japan

FIGURE 5.11: Baseline Japan-to-NSF / NSF-to-Japan - Boxplots, Accuracy comparison

5.4.1 Random Forest

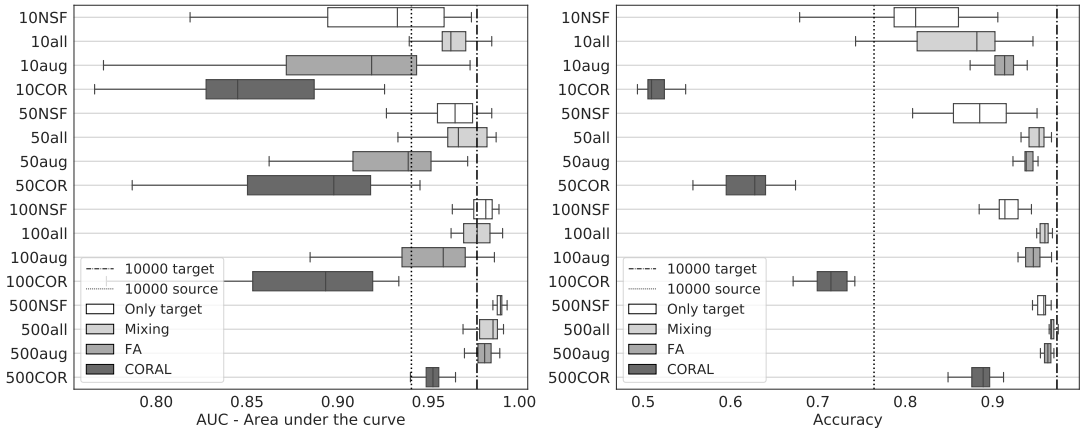
We start our analysis considering the Random Forest classifier used in the previous cases, but this time, we integrated the experiments with the transfer learning models.

Figure 5.12 shows the performance achieved by the classifier under the **Japan-to-NSF** scenario, in particular Figure 5.12a reports the AUC performance values. It is possible to note that in this case, the highest performance are reached by the *mixing* setting, while FA and CORAL approaches, lightgrey and darkgrey boxplots, respectively, obtain discretely good values of AUC, that increase with the increasing of the number of instance.

Although the AUCs are around 0.9, the two domain adaptation techniques do not outperform the other approaches.

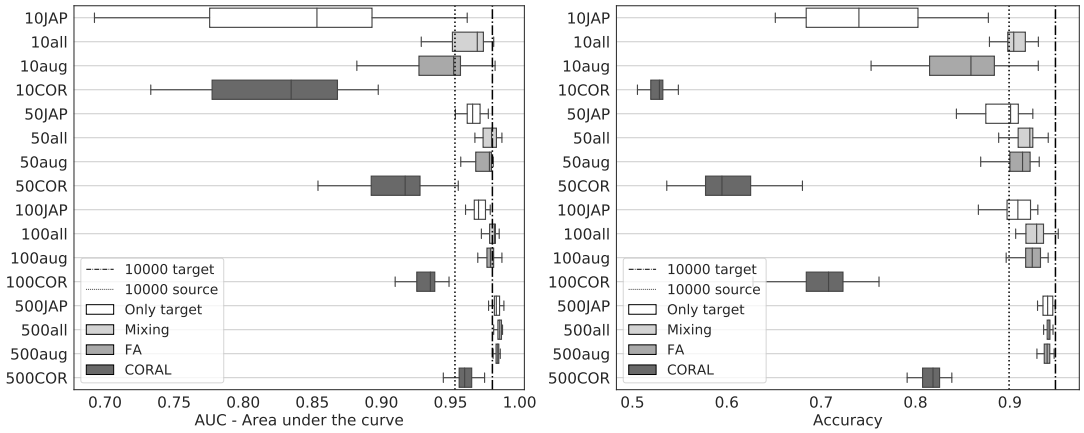
Figure 5.12b represents instead the results related to Accuracy metric; in this case the values are slightly lower than AUCs, but the trend is equal to the previous one, except for CORAL that instead, achieves lower result for 10, 50 and 100 samples and acceptable values (but not as high as the others), improvements for 500 samples. Also in this case we do not achieve performance exploiting the two DA approaches, except for the single case of 10aug, Figure 5.12b, which is slightly larger than 10NSF and 10all cases.

We replicated the same experiments considering **NSF-to-Japan**; results are shown in Figure 5.13, where in general AUC values are slightly larger than Accuracy values. Also for this case, both Feature Augmentation and CORAL are always below the *mixing* setting and the *only target* setting.



(A) AUC - Random Forest (B) Accuracy - Random Forest

FIGURE 5.12: Domain adaptation Japan-to-NSF & Random Forest



(A) AUC - Random Forest (B) Accuracy - Random Forest

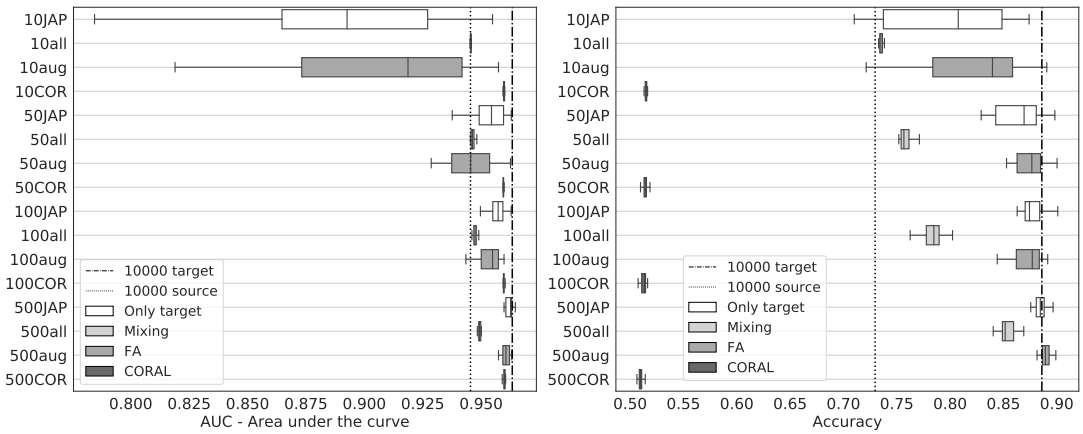
FIGURE 5.13: Domain adaptation NSF-to-Japan & Random Forest

Results for these two scenarios show that adoption of TL models does not provide a valid alternative to the simple mixing of the source data with target data. For this reason, we decided to investigate alternative ML models that can perform the classification. So, motivated by the use of the same two TL techniques, in the following we investigate on the exploitation of different learning algorithms: i.e., the Support Vector Machine and Logistic Regression.

5.4.2 Support Vector Machine (SVM)

As said, we decided to move from a Random Forest classifier to a Support Vector Machines algorithm described in Section 3.1.3, which are used to perform the same classification problem. In order to be consistent with respect to the previous case, we perform the same experiments done before, first analyzing the **NSF-to-Japan** scenario the opposite one.

Figure 5.15a plots the resulting AUC in the specific case where NSF network is used

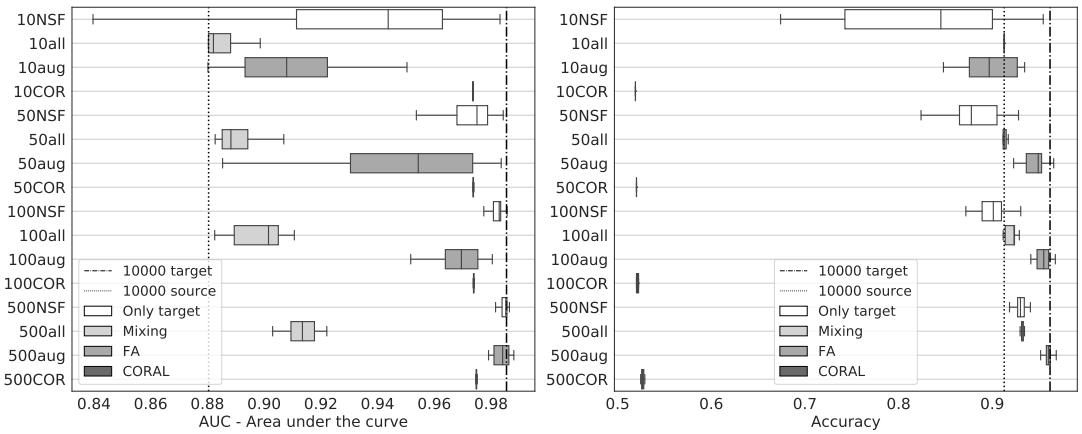


(A) AUC - Support Vector Machine (B) Accuracy - Support Vector Machine

FIGURE 5.14: Domain adaptation NSF-to-Japan & Support Vector Machine

as source domain, while Japan network is adopted as target domain. The average lightpath length in the NSF topology (See Fig. 5.2) is consistently higher than in he Japan topology (See Fig. 5.1). For a deep inspection on that, it is possible to compare the distributions of lightpath length reported in B for the case in Fig. B.2a, where the lightpaths deployed in in the Japan network never exceed 2000 km, whereas in the NSF network the maximum reached is around 8000 km, though lightpath lengths in the order of hundreds of km there exist also for this topology. Therefore, the train source set $\mathcal{R}_{NSF}$ (or S) is expected to contain a representative number of lightpath samples for a wide range of lengths. By this fact, there exist a small gap, around 0.02, between the AUC values obtained in *10000 source* and *10000 target* scenarios, meaning that algorithm which learns from samples gathered from the NSF already provides a reasonable knowledge on the BER of lightpaths deployed in Japan network. In addition, when the sampled train target set $T \subseteq \mathcal{R}_{Japan}$ is low, *10000 source* scenario outperforms *only target* scenario; this example confirms that learning from a high number of train source set S samples is more effective than learn from a limited number of train target set T. Moreover, learning from the set $S \cup T$, i.e., the *mixing* setting, leads to modest improvements on the AUC values when T is high. As for the transfer learning approaches, when T is low, CORAL significantly improves the AUC values with respect to the *10000 source* scenario, *only target* setting and *mixing* setting, while Feature Augmentation achieves on average lower AUC than *mixing* setting and manifests comparable performance to *only target* setting. Conversely, when $T = 500$, the *only target*, FA and CORAL provide similar results, which closely approach *10000 target* cases. Indeed, it is possible to conclude that, when the number of available samples from the target domain train set is quite large, DA techniques are expected to be less useful, as those samples are already representative of the feature space.

In Figure 5.15a we report the AUC results in the case we used Japan topology as



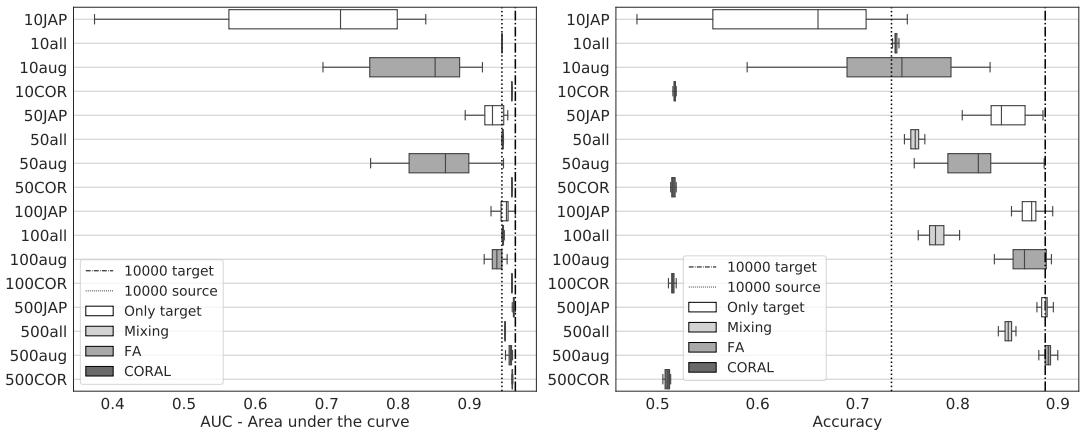
(A) AUC - Support Vector Machine (B) Accuracy - Support Vector Machine

FIGURE 5.15: Domain adaptation Japan-to-NSF & Support Vector Machine

source domain and NSF topology as target domain. In this case, the gap between the *10000 source*, i.e., only 10000 samples of Japan network, and *10000 target*, i.e., only 10000 samples of NSF network, is larger than before (Fig 5.15a). As the average lightpath length in the Japan topology is considerably lower with respect to that of NSF topology, little knowledge about the BER of long lightpaths can be obtained through the samples of the Japan network, i.e., the source domain used to compute *10000 source*. For this reason, even when T is low, the *only target* almost always outperform *10000 source*. This could be an hint, that learning from a few samples gathered from the NSF network yields to knowledge about the lightpaths for which the links are long, rather than relying on a large amount of samples of short lightpaths obtained from the source domain. Indeed considering the performance of *mixing* cases, it turns out that learning from $S \cup T$, always leads to worse AUC values, with respect to learning only from T. For what concern the DA approaches, both FA and CORAL outperform the *10000 source* scenario and the mixing cases. In Figure 5.15a, CORAL shows highest AUC values for small T, whereas, Feature Augmentation shows comparable performance to the *only target* cases and more closely approaches the baseline *10000 target* than CORAL.

Figure 5.15b presents the analysis of the Accuracy, which globally shows lower levels w.r.t the AUC. In particular, CORAL manifests a constant behavior around 0.5, while FA performs well, reaching both with few samples and also with many samples of target domain the higher values of the *only target* and the *best case* setting. The *mixing*, instead resides in the middle of CORAL and FA, presenting a notable increase with many sample of target. Figure 5.14b instead is comparable with 5.15b especially for the different gap between *best* and *worse* scenario, which results more restricted. CORAL behaves wrong as before, while the *mixing* manifests good performance already from few samples of target.

We now move our considerations quantifying, at least visually, how much a different



(A) AUC - Support Vector Machine (B) Accuracy - Support Vector Machine

FIGURE 5.16: Domain adaptation NSF-to-Japan with features subset S1 & Support Vector Machine

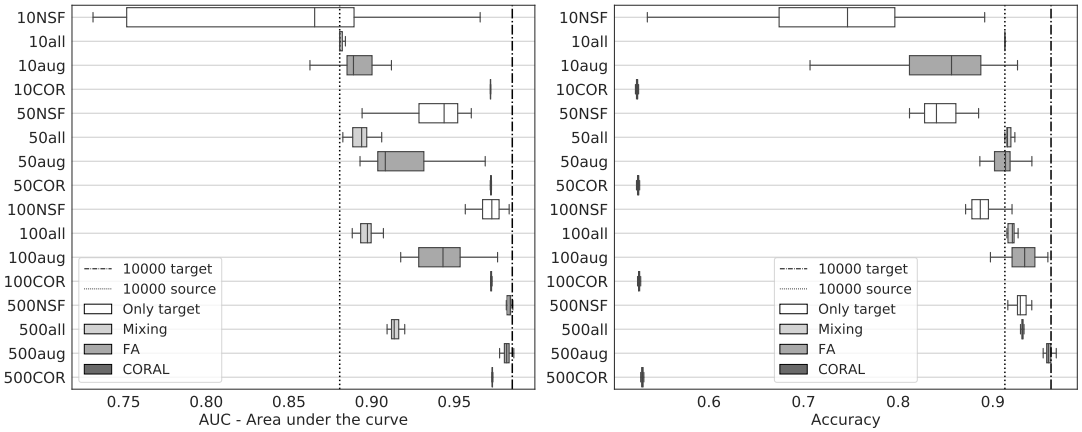
feature selection impacts the performance in the cases described above. In particular we shift our selection from the subset $S2$, i.e., with the selection of the features: *number of links*, *lightpath length*, *length of longest link*, *traffic volume* and *modulation format*, to the subset $S1$, which is the same as $S2$, but extended with six additional features (See Table 5.3).

As before, we reported the results in terms of AUC and Accuracy, related to the **NSF-to-Japan** and **Japan-to-NSF** scenarios. Figure 5.16 and 5.17 can be easily compared with Figure 5.14 and 5.15 respectively. It is possible to note that for each scenario the trends followed by the AUC values are very similar.

The good aspect is that also with this selection of features the TL techniques outperform in many cases both the *only target* and the simple *mixing*. CORAL always beats the *10000 source* baseline, while FA outperforms the 10000 source baseline where T is large enough, e.g., 100 or 500 samples. Note that, using a total amount of 11 features increases the computational time to perform the training because the feature space has more dimensions.

Furthermore, this leads to an increase of the variance in the distributions of the experiments results, (that are replicated 20 times). So, with an expanded feature space, the variance of the distribution of the AUC values increases.

Finally, we have evaluated the scenario in which Japan* network with different parameter of loss is tested with NSF network first as source source domain, then as target domain. In this particular cases it is possible to appreciate the same trend of the Figure 5.14 and Figure 5.15, remarking the higher effectiveness of the TL techniques, Feature Augmentation and CORAL, w.r.t the other techniques. Although the trend is similar to previous cases we observed, generally, a slightly deterioration of the performance, i.e., AUC and Accuracy are generally lower for all the settings. This is a direct consequence of the diverse penalty terms setting, modified during the



(A) AUC - Support Vector Machine (B) Accuracy - Support Vector Machine

FIGURE 5.17: Domain adaptation Japan-to-NSF with features subset S1 & Support Vector Machine

generation of the dataset, indeed the attenuation of the fiber have been augmented by 0.05 dB/Km and the noise figure amplifier lifted from 5 to 6 dB.

5.4.3 Logistic Regression

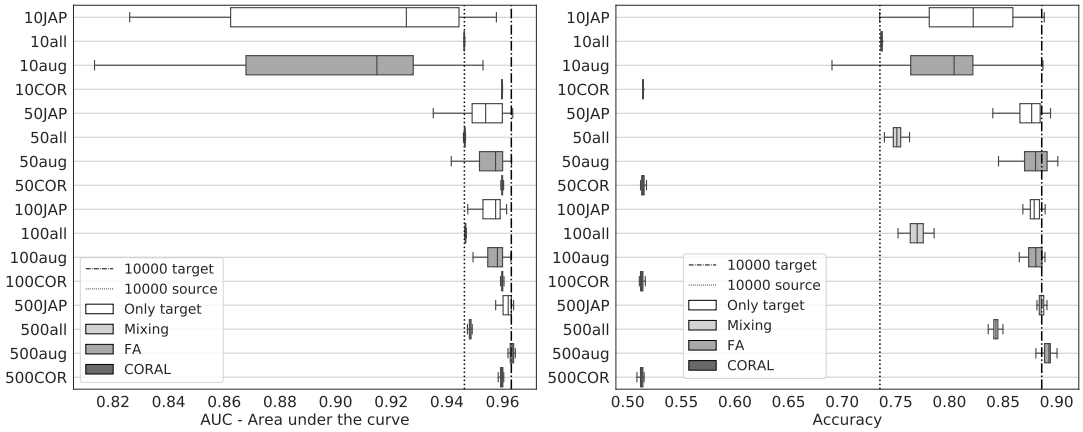
As last step for the experiment that regards the Domain Adaptation assessment, we implemented also an additional classification model, i.e. a generalized linear model, although it is based on a decision rule that is carried out from a *sigmoid function*: the *Logistic Regression* (See 3.1.3 for details). For these set of results we considered the same scenarios of the SVM classifier used before.

Figure 5.18 and 5.19 illustrate the results achieved by AUC and Accuracy under the **NSF-to-Japan** scenario and **Japan-to-NSF**.

In particular it is possible to verify that for the first scenario, the two trends follow in almost the totality of settings, the behavior already seen for the SVM classifier in Figure 5.14. Indeed, it is true that the average lightpath length in NSF topology is higher than in the Japan one, thus a little portion of the two supports is shared, and for this reason, as said before, the train set S is expected to include many lightpath samples for a wide range of lengths.

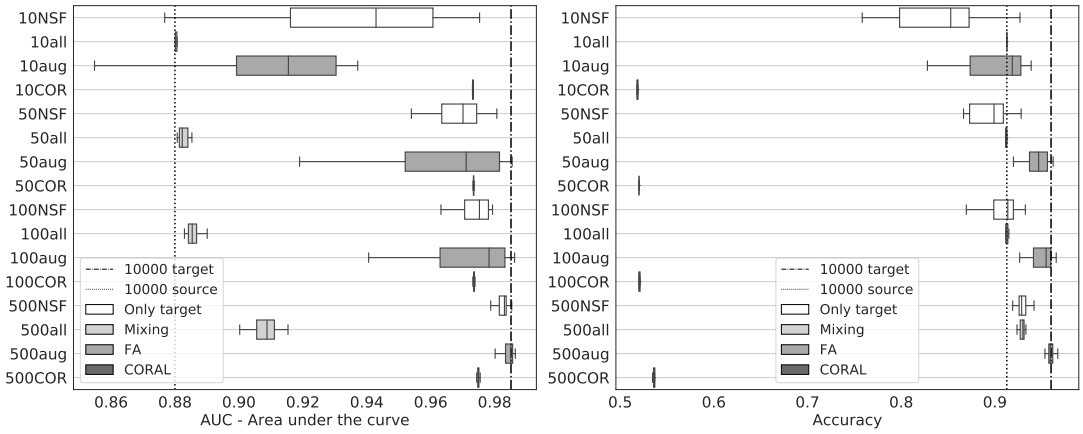
In such way the samples collected from NSF topology and used to train the model provide a reasonable knowledge on BER of the lightpaths that could be find in Japan topology.

Furthermore, the case in which the sampled target set T is low, shows that *10000 source* baseline outperforms *only target* setting and this confirms that learning from a numerous source set S is more effective than learning from a limited number of target set T.



(A) AUC - Logistic Regression (B) Accuracy - Logistic Regression

FIGURE 5.18: Domain adaptation NSF-to-Japan & Logistic Regression



(A) AUC - Logistic Regression (B) Accuracy - Logistic Regression

FIGURE 5.19: Domain adaptation Japan-to-NSF & Logistic Regression

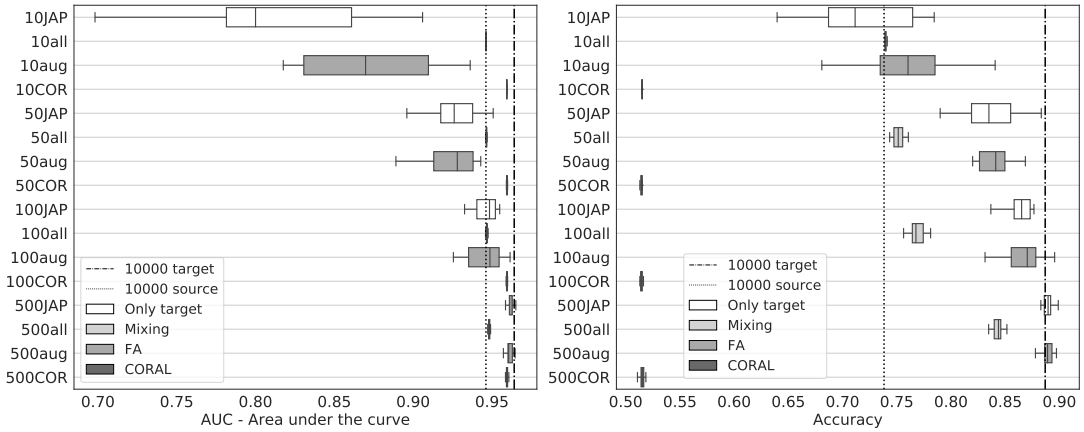
When T is low, the performance of CORAL grows up w.r.t the *10000 source* baseline, *only target* setting and *mixing*. FA instead is lower than than *mixing* setting and presents a trend comparable to that of the *only target* setting.

When T is high, instead, *only target*, FA and CORAL hold similar trends that approach the *10000 target* baseline.

Finally it can be asserted that when the number of samples from target domain T is quite large, the TL techniques results less useful because those samples are already representative of the feature space.

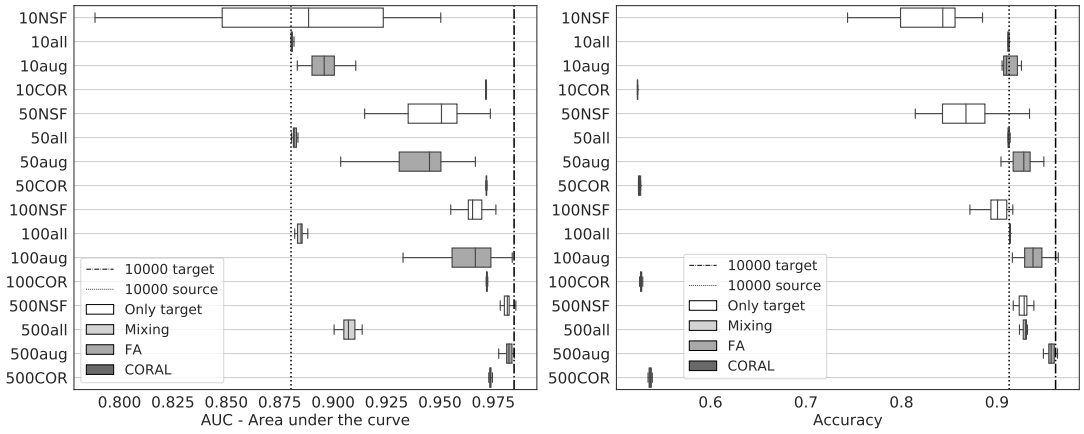
Also for the second scenario (**Japan-to-NSF**), the pair of plots show a similar situation with respect to that of SVM.

The cases in which the S1 subset (containing 11 features) is adopted are shown in Figure 5.20 and 5.21. Also for this case the performance achieved by logistic



(A) AUC - Logistic Regression (B) Accuracy - Logistic Regression

FIGURE 5.20: Domain adaptation NSF-to-Japan with features subset S1 & Logistic Regression



(A) AUC - Logistic Regression (B) Accuracy - Logistic Regression

FIGURE 5.21: Domain adaptation Japan-to-NSF with features subset S1 & Logistic Regression

regression are similar to SVM with a very little decrement in only few cases regarding the scenario **NSF-to-Japan**.

With logistic regression we are able to confirm the effectiveness of our proposed DA techniques, which exploit the feature space transformation, and we are motivated to proceed in exploring particular insights relative to critical cases described in the following. In particular we are going to explore the testing made using samples which are difficult to classify because have a BER value which is near to the decision threshold.

5.5 Impact of dissimilarities between lightpath distributions

In this last section, we try to put in relation the previous results with a similarity measure, which is able to verify that the trends of performance are directly linked

to the grade of similarity between the domains at which traditional ML and TL techniques have been applied.

A first approach to verify the statement could be to quantify the overlaps of considered pairs of distributions (See Appendix A) using the intersection metric.

5.5.1 Intersection metric

The intersection (I) between two probability distribution functions (pdfs) have been widely used as a form of similarity. It is defined as [4]:

$$I = \sum_{i=0}^{d} \min(P_i, Q_i) \tag{5.3}$$

where P and Q correspond to the pdf values of the distribution used as source domain and as a target domain, respectively, while d represents the length of the two vectors.

In order to graphically evaluate the relation among intersection metric and achieved performance, we considered the case in which Support Vector Machine with *S2* subset of features were adopted, using the area under the curve (AUC) as performance metric (See Figure 5.14 and 5.15).

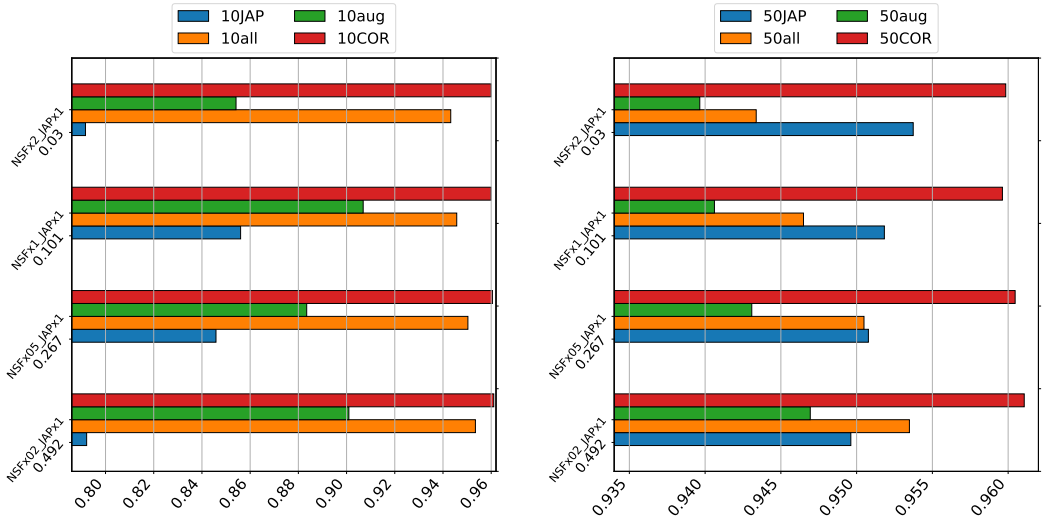
In particular, we replicated the same experiments for all the combinations of networks, grouping results in relation to the same *target* domain network. So for example, if the pair *source-target* were Japan network-NSF network, we grouped the same NSF network for all the re-scaled Japan topologies and so on.

We reported the resulting scenarios of the experiments run, considering both the **NSF-to-Japan** and **Japan-to-NSF**, and the corresponding cases with a different number of samples (10, 50, 100, 500) of the *target* domain.

Figures 5.22 and 5.23 illustrate the comparison among intersection metric (I) and performance achieved in the **NSF-to-Japan**.

It is possible to note that the trends are coherent with the metric values, indeed, the more the two networks overlap, the more the performance increase. In particular, the *mixing* (yellow bars) scenario manifests a degradation w.r.t the decrease of intersection I value, i.e., when the grade of dissimilarity of the two networks increase.

We can observe that the two DA techniques (red and green bars) outperforms the *mixing*; CORAL beats *mixing* already when few samples of target are considered (Figure 5.22a, 5.22b, 5.23a, 5.23b), while FA outperforms *mixing* when the samples of target are more, i.e., 100 and 500 (Figure 5.23a and 5.23a). Furthermore, both DA approaches often outperform the *only target* setting (blue bars) especially with few samples.



(A) AUC NSF-to-JAP / Intersection metric vs 10 target samples

(B) AUC NSF-to-JAP / Intersection metric vs 50 target samples

FIGURE 5.22: Intersection metric in NSF-to-Japan scenario, 10 and 50 target samples

It is also possible to note a constant trend of CORAL almost in the entire cases (10, 50, 100 and 500 target samples), while FA is more sensitive to the increase of the target samples.

In conclusion, the worsening of DA techniques performance with the increase of the degree of dissimilarity is less pronounced with respect to the *mixing* setting, in some case it does not exist, especially with CORAL (5.23a and 5.22b).

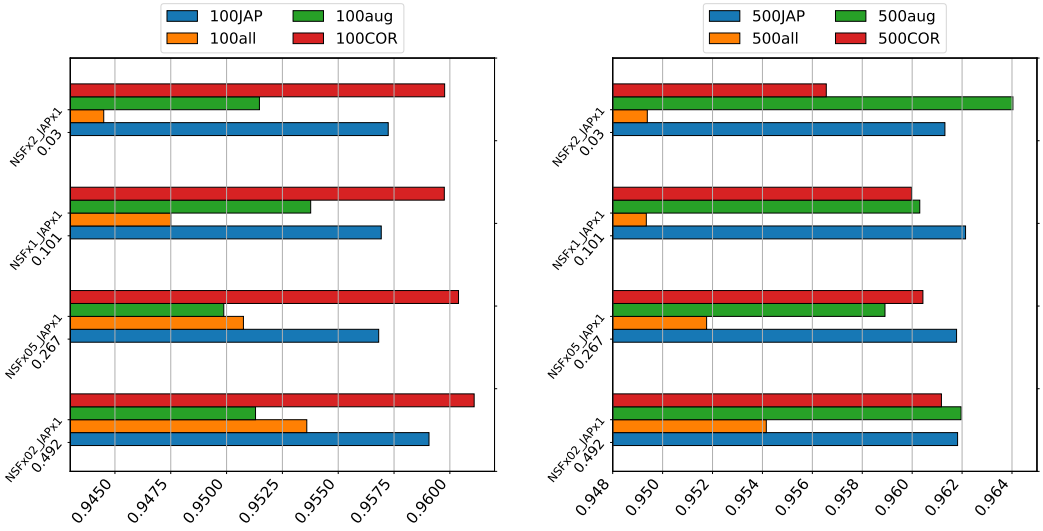
Figure 5.24 and 5.25 illustrate the comparison among intersection metric (I) and performance achieved in the **Japan-to-NSF**.

As in the previous case, the trends obtained grouping the NSF topology with scaling factor $SF = 1$ are coherent with the metric values: the more the two networks overlap, the more the performance improves.

Indeed, from the first case in which only 10 samples of target are considered, to that in which 500 target samples are used, it is evident that performance degrade with the decrease of intersection value I (Figure 5.24a, 5.24b, 5.25a and 5.25b).

The decreasing trend of *mixing* (yellow bars) is more accentuated w.r.t that of the two DA approaches. FA and CORAL present generally higher values and always outperform the *mixing* setting.

An evident difference with respect to the previous case (Figure 5.22 and 5.23), is that *only target* setting presents higher values, so it is not easy for FA and CORAL to beat the blue bars, especially in the cases where the number of target samples is high (Figures 5.24b, 5.25a and 5.25b). Generally, in this case, we can observe a constant



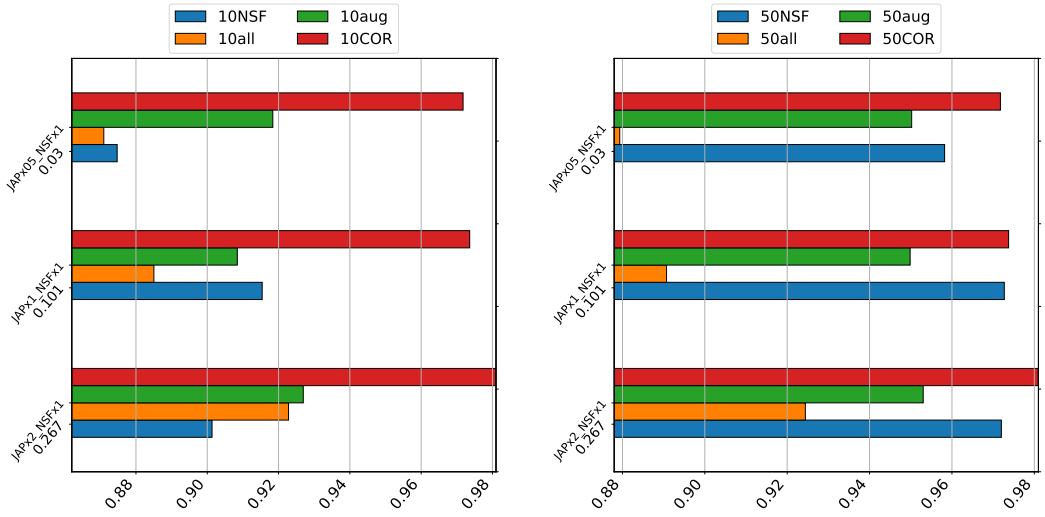
(A) AUC NSF-to-JAP / Intersection metric vs 100 target samples

(B) AUC NSF-to-JAP / Intersection metric vs 500 target samples

FIGURE 5.23: Intersection metric in NSF-to-Japan scenario, 100 and 500 target samples

behavior of CORAL and *mixing* trends in all the four cases, but CORAL and FA are less sensitive to the decreasing of the intersection metric.

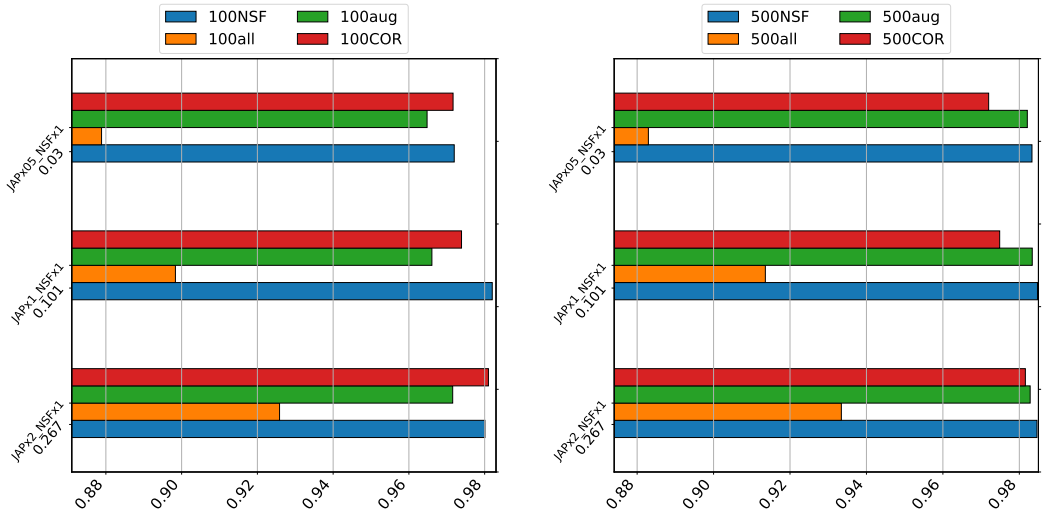
These results generalize the previous numerical and graphical analysis, highlighting the usefulness of DA techniques for QoT estimation, that in many scenarios have been shown to ouperform with respect to traditional ML algorithms.



(A) AUC Japan-to-NSF / Intersection metric vs 10 target samples

(B) AUC Japan-to-NSF / Intersection metric vs 50 target samples

FIGURE 5.24: Intersection metric in Japan-to-NSF scenario, 10 and 50 target samples



(A) AUC Japan-to-NSF / Intersection metric vs 100 target samples

(B) AUC Japan-to-NSF / Intersection metric vs 500 target samples

FIGURE 5.25: Intersection metric in Japan-to-NSF scenario, 100 and 500 target samples

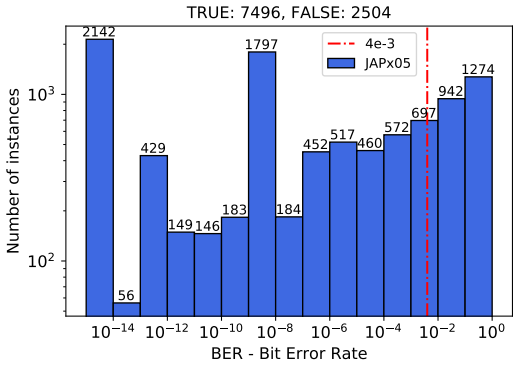
Chapter 6

Conclusion

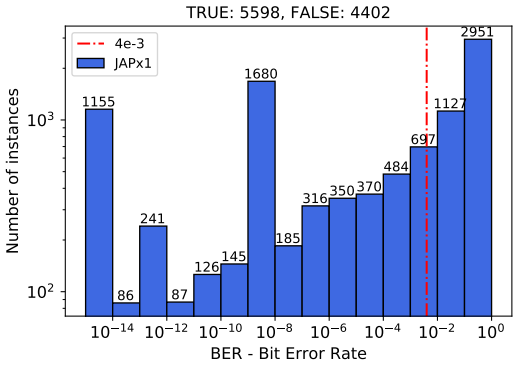
The ever increasing traffic volumes served by backbone optical networks require the development of transmission systems which can ensure suitable QoT requirements. For this reason, the prediction of the QoT of a candidate lightpath plays a pivotal role for an effective design and management of optical networks. In the last few years, ML have found a lot of applications as technique for the QoT estimation in optical networks. In this thesis, the particular case where QoT has to be predicted for a different, but related domain has been investigated. Unfortunately, for new deployed optical systems, the availability of data is not sufficient or the acquisition of large datasets requires high costs. For this reason, it is necessary to optimize the utilization of few data of target domain and integrate with the large datasets of the source domain. Specifically, in this thesis we have focused on domain adaptation approaches: DA techniques transform the features space, either augmenting or aligning the source and target domains. We assessed the performance of two type of DA techniques in different settings, where we change the number of samples belonging to the target domain. Achieved results show that Feature Augmentation [10] and CORAL [19] are able to outperform the basic mixing of the data. In particular, it has been seen that training a Support Vector Machine classifier with data for which have been applied DA approaches, it is possible to develop models which reach values of area under the curve (AUC) and Accuracy around 0.9, exploiting no more than 500 samples of target domain. In addition, we replicated same experiments, changing the ML model, i.e., using the logistic regression, and we obtained almost same results, confirming the improvements in terms of performance provided by Domain Adaptation techniques. Finally, the *intersection metric* has been adopted to explore the dependency of the AUC on the degree of similarity between the two distributions of lightpath lengths of the two domains. Results show that DA techniques reduce the AUC when the intersection between the two distributions becomes smaller.

Appendix A

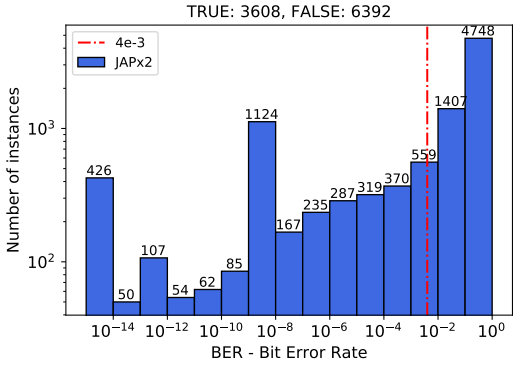
Bit Error Rate - BER distributions



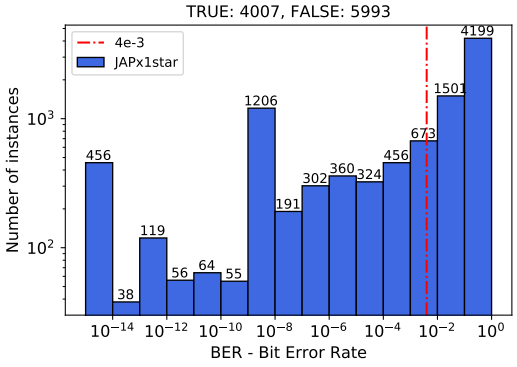
(A) BER distribution in Japan topology, SF=0.5



(B) BER distribution in Japan topology, SF=1

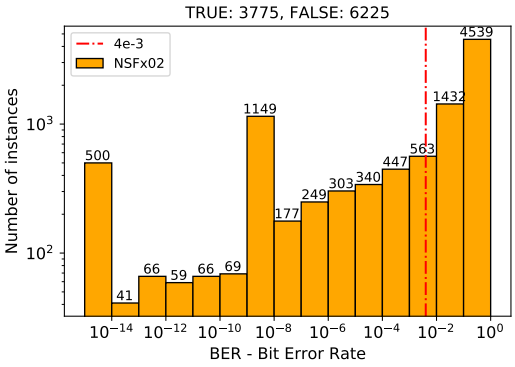


(C) BER distribution in Japan topology, SF=2

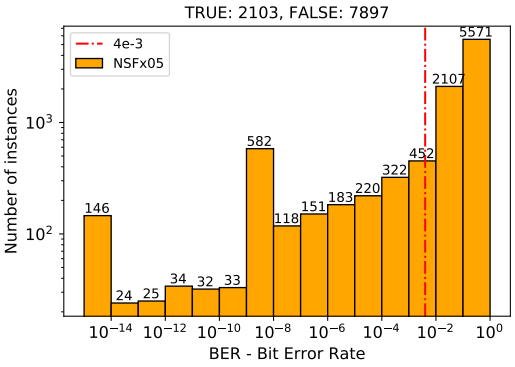


(D) BER distribution in Japan$*$ topology, SF=1

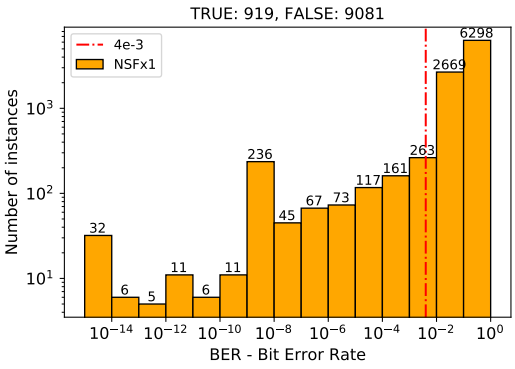
FIGURE A.1: Logarithmic histograms of BER distribution - Part I



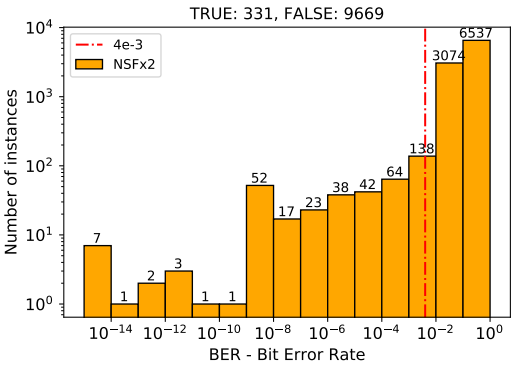
(A) BER distribution in NSF topology, SF=0.2



(B) BER distribution in NSF topology, SF=0.5



(C) BER distribution in NSF topology, SF=1

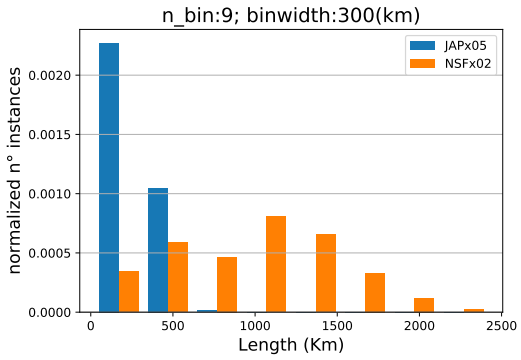


(D) BER distribution in NSF topology, SF=2

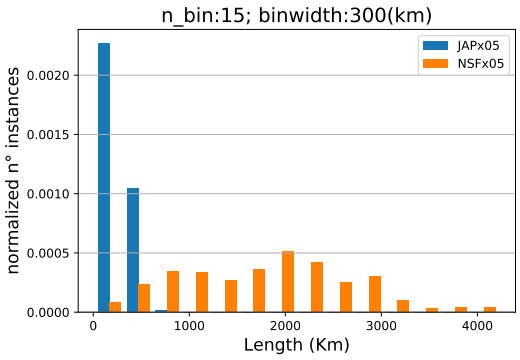
FIGURE A.2: Logarithmic histograms of BER distribution - Part II

Appendix B

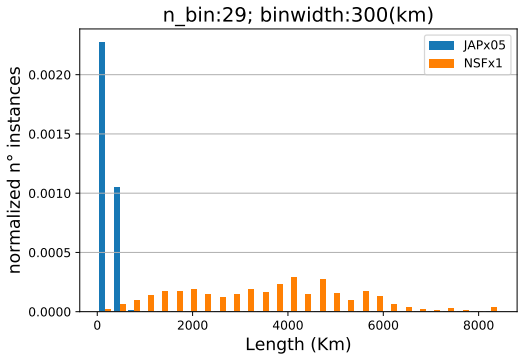
Lightpath length distributions



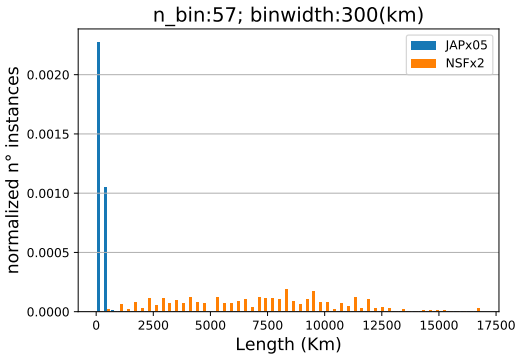
(A) Lightpath length distribution in Japan topology, SF=0.5 & NSF topology, SF=0.2



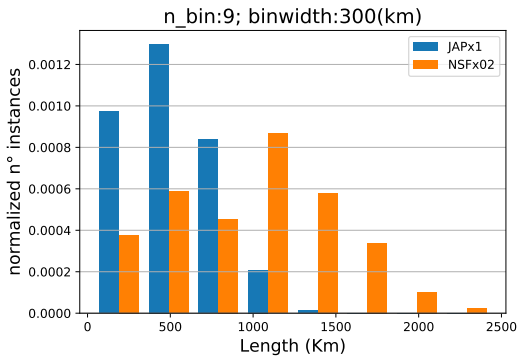
(B) Lightpath length distribution in Japan topology, SF=0.5 & NSF topology, SF=0.5



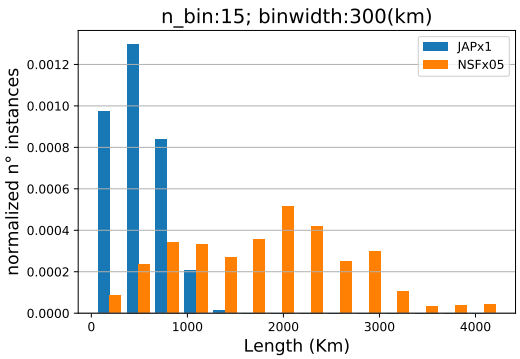
(C) Lightpath length distribution in Japan topology, SF=0.5 & NSF topology, SF=1



(D) Lightpath length distribution in Japan topology, SF=0.5 & NSF topology, SF=2

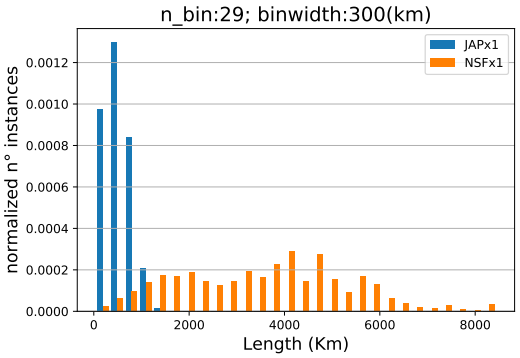


(E) Lightpath length distribution in Japan topology, SF=1 & NSF topology, SF=0.2

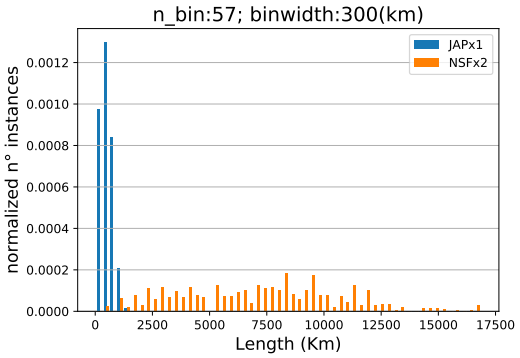


(F) Lightpath length distribution in Japan topology, SF=1 & NSF topology, SF=0.5

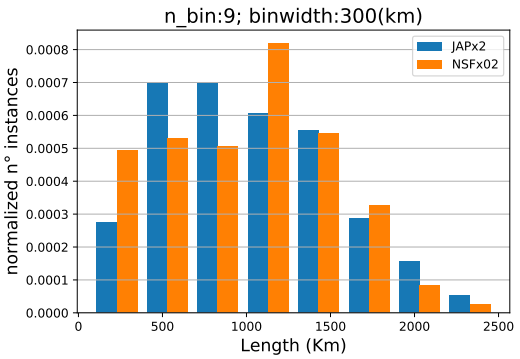
FIGURE B.1: Distributions of $\mathcal{R}$ lightpath lenghts - Part I



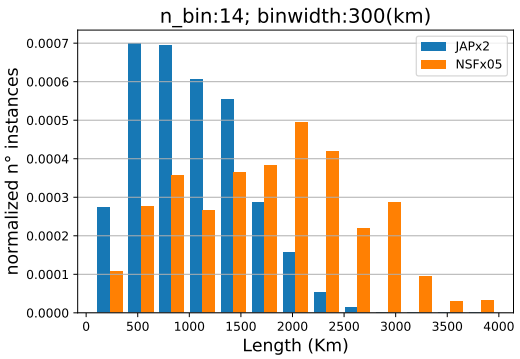
(A) Lightpath length distribution in Japan topology, SF=1 & NSF topology, SF=1



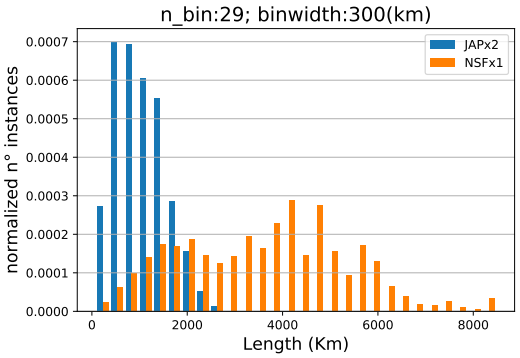
(B) Lightpath length distribution in Japan topology, SF=1 & NSF topology, SF=2



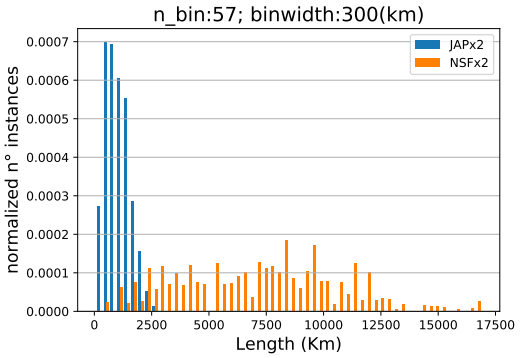
(C) Lightpath length distribution in Japan topology, SF=2 & NSF topology, SF=0.2



(D) Lightpath length distribution in Japan topology, SF=2 & NSF topology, SF=0.5



(E) Lightpath length distribution in Japan topology, SF=2 & NSF topology, SF=1



(F) Lightpath length distribution in Japan topology, SF=2 & NSF topology, SF=2

FIGURE B.2: Distributions of $\mathcal{R}$ lightpath lenghts - Part II

Bibliography

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006, pp. 1–58.

[2] G. Bosco et al. "On the Performance of Nyquist-WDM Terabit Superchannels Based on PM-BPSK, PM-QPSK, PM-8QAM or PM-16QAM Subcarriers". In: *Journal of Lightwave Technology* 29.1 (2011), pp. 53–61. DOI: 10.1109/JLT.2010.2091254.

[3] Andrea Carena et al. "EGN model of non-linear fiber propagation". In: *Optics express* 22 (June 2014), pp. 16335–16362. DOI: 10.1364/OE.22.016335.

[4] Sung-Hyuk Cha. "Comprehensive Survey on Distance/Similarity Measures Between Probability Density Functions". In: *Int. J. Math. Model. Meth. Appl. Sci.* 1 (Jan. 2007).

[5] Ronen Dar et al. "Accumulation of nonlinear interference noise in fiber-optic systems". In: *Optics express* 22 (Oct. 2013). DOI: 10.1364/OE.22.014199.

[6] Basura Fernando et al. "Unsupervised Visual Domain Adaptation Using Subspace Alignment". In: (Dec. 2013). DOI: 10.1109/ICCV.2013.368.

[7] Peter Flach, Jose Hernandez-Orallo, and Cèsar Ferri. "A Coherent Interpretation of AUC as a Measure of Aggregated Classification Performance." In: *Proceedings of the 28th International Conference on Machine Learning, ICML 2011* (Jan. 2011), pp. 657–664.

[8] Maayan Harel and Shie Mannor. "Learning from Multiple Outlooks". In: *Computing Research Repository - CORR* (Apr. 2010).

[9] Computational resources provided by HPC@POLITO. *which is a project of Academic Computing within the Department of Control and Computer Engineering at the Politecnico di Torino (http://hpc.polito.it)*.

[10] Hal III. "Frustratingly Easy Domain Adaptation". In: *ACL 2007 - Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics* (July 2009).

[11] Weiyang Mo et al. "ANN-Based Transfer Learning for QoT Prediction in Real-Time Mixed Line-Rate Systems". In: (Jan. 2018), W4F.3. DOI: 10.1364/OFC.2018.W4F.3.

[12] Kevin P. Murphy. *Machine Learning : a Probabilistic Perspective*. Mass.: MIT Press, 2012, pp. 21–22.

[13] F. Musumeci et al. "An Overview on Application of Machine Learning Techniques in Optical Networks". In: *IEEE Communications Surveys Tutorials* 21.2 (2019), pp. 1383–1408. DOI: 10.1109/COMST.2018.2880039.

[14] Francesco Musumeci et al. "A Tutorial on Machine Learning for Failure Management in Optical Networks". In: *Journal of Lightwave Technology* PP (June 2019), pp. 1–1. DOI: 10.1109/JLT.2019.2922586.

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

[16] Cristina Rottondi et al. "Machine-Learning Method for Quality of Transmission Prediction of Unestablished Lightpaths". In: *Journal of Optical Communications and Networking* 10 (Feb. 2018), A286. DOI: 10.1364/JOCN.10.00A286.

[17] Matteo Salani, Cristina Rottondi, and Massimo Tornatore. "Routing and Spectrum Assignment Integrating Machine-Learning-Based QoT Estimation in Elastic Optical Networks". In: Apr. 2019, pp. 1738–1746. DOI: 10.1109/INFOCOM.2019.8737413.

[18] "Spectral grids for WDM applications: DWDM frequency grid". In: *ITU-T Reccomendation G.694.1* (Feb. 2012).

[19] Baochen Sun, Jiashi Feng, and Kate Saenko. "Return of Frustratingly Easy Domain Adaptation". In: (Nov. 2015).

[20] Le Xia et al. "Transfer learning assisted deep neural network for OSNR estimation". In: *Optics Express* 27 (July 2019), p. 19398. DOI: 10.1364/OE.27.019398.

[21] Qiuyan Yao et al. "Spectrum Optimization for Resource Reservation Based on Transductive Transfer Learning in Space Division Multiplexing Elastic Optical Networks". In: (Sept. 2018), pp. 1–3. DOI: 10.1109/ECOC.2018.8535242.

[22] J. Yu et al. "Model transfer of QoT prediction in optical networks based on artificial neural networks". In: *IEEE/OSA Journal of Optical Communications and Networking* 11.10 (2019), pp. C48–C57. DOI: 10.1364/JOCN.11.000C48.

[23] X. Zhou et al. "High Spectral Efficiency 400 Gb/s Transmission Using PDM Time-Domain Hybrid 32–64 QAM and Training-Assisted Carrier Recovery". In: *Journal of Lightwave Technology* 31.7 (2013), pp. 999–1005. DOI: 10.1109/JLT.2013.2243643.