FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Adversarial Domain Adaptation for Sensor Networks

Francisco Tuna Andrade

Mestrado Integrado em Engenharia Informática e Computação

Supervisors: Jaime Cardoso and Diogo Pernes

July 5, 2020

Adversarial Domain Adaptation for Sensor Networks

Francisco Tuna Andrade

Mestrado Integrado em Engenharia Informática e Computação

July 5, 2020

Abstract

With the recent surge of big data, manually annotating datasets has become an impossible task. As a result, domain adaptation has gained more importance than ever, since it allows the transfer of knowledge from a labeled dataset to an unlabeled one. That is even truer for sensor networks, where the domain shifts between different sensors are usually substantial and models developed with a set of source sensors fail to generalize well to new target sensors.

Adversarial neural networks, which were introduced in 2014 in the scope of generative adversarial neural networks, are a promising tool to address the domain adaptation problem. Despite being a recent idea, adversarial networks have already led to tremendous progress in multiple areas of the field. In this dissertation, we propose new adversarial domain adaptation models, specifically tuned for sensor networks, that make use of Long short-term memory (LSTM) networks, both for discriminating between source and target domains and for performing the learning task. The fact that we take the temporal nature of the data into account by using an LSTM network for the discriminating task is a novel idea. We evaluate the performance of our models and compare them with other methods by conducting extensive experiments that demonstrate their robustness and value. In particular, DoubleLSTM, a model proposed by us, showed an average improvement of 27% for the Mean Absolute Error (MAE) on an object counting task, when compared to other state-of-art models.

Besides the proposed models, we include a survey of domain adaptation methods. Additionally, we also discuss several open lines of research in the field of adversarial domain adaptation that can serve as a guide for future work in this area.

Keywords: Domain adaptation, Adversarial neural networks, Sensor networks, LSTM networks

Resumo

Com o recente surgimento da Big Data, anotar manualmente datasets tornou-se uma tarefa impossível. Como resultado, adaptação de domínio ganhou acrescida importância, por permitir a transferência de conhecimento de um dataset etiquetado para outro não etiquetado. Tal é ainda mais relevante para redes de sensores, onde as diferenças de domínio entre diferentes sensores são substanciais e modelos treinados com um conjunto de sensores têm dificuldade em generalizar bem para novos sensores.

As redes neuronais adversariais, introduzidas em 2014, são uma ferramenta promissora para abordar o problema da adaptação de domínio. Apesar de serem uma ideia recente, as redes adversariais deram origem a progressos notáveis em múltiplas vertentes. Nesta dissertação, propomos novos modelos adversariais para adaptação de domínio, adequados especialmente para redes de sensores, que utilizam redes do tipo LSTM (Long Short-Term Memory) para discriminar entre a fonte e o domínio ou para executar a tarefa de aprendizagem. O facto de termos a natureza temporal dos dados em consideração ao usarmos uma rede LSTM para a tarefa discriminante é uma ideia não explorada anteriormente. Avaliamos o desempenho dos nosso modelos e comparamo-los com outros métodos, conduzindo diversas experiências que demonstram a sua robustez e o seu valor. Em particular, o modelo DoubleLSTM, proposto por nós, mostrou uma melhoria de 27% no erro médio absoluto numa tarefa de contar objetos, quando comparado com outros modelos do estado da arte.

Para além dos modelos propostos, incluímos um estudo de métodos de adaptação de domínio. Adicionalmente, também discutimos linhas de investigação em aberto no campo de adaptação de domínio que servem como um guia para futuros trabalhos nesta área.

Keywords: Adaptação de domínio, Redes neuronais adversariais, Redes de sensores, Redes LSTM

Acknowledgements

I would like to thank PhD student Diogo Pernes for guiding me in this dissertation and having the patience for reviewing my dissertation report countless times.

I would also like to thank Professor Dr. Jaime Cardoso for proposing the topic of this dissertation and the advice he gave me.

Lastly, I want to thank the entire Visual Computing and Machine Learning group at INESC-TEC Porto for supporting the work I developed.

Francisco Andrade

"People worry that computers will get too smart and take over the world, but the real problem is that they're too stupid and they've already taken over the world."

Pedro Domingos

Contents

List of Figures

List of Tables

Abbreviations

CNN	Convolutional Neural Network
CORAL	Correlation Alignment
DA	Domain Adaptation
DANN	Domain Adaptation Neural Network
GAN	Generative Adversarial Network
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MDAN	Multisource Domain Adversarial Networks
MMD	Maximum Mean Discrepancy
TL	Transfer Learning

Chapter 1

Introduction

This chapter provides an overview of the dissertation. It starts by providing a context, where some important terms are defined. Afterwards, the motivation for this dissertation is introduced and the goals we hope to achieve are enumerated. Lastly, it presents the overall structure of this document.

1.1 Context

Supervised machine learning models demand the use of large and labeled datasets. As a consequence, collecting data has become as important as designing the models. Unfortunately, collecting and annotating such large datasets is expensive and often even flat out impossible. Domain adaptation (DA) surges as an answer to this issue by allowing the transfer of knowledge from a labeled source domain to an unlabeled target domain.

Several methods have been proposed to address domain adaptation. Reweighting algorithms seek to reweight the source distribution so that it looks like the target one [17], [18]. Iterative methods train a model with the labeled samples and classify the unlabeled target ones. Afterwards, a new model is trained with the now labeled target samples. This process is then repeated several times [19], [20]. In recent years, adversarial approaches, where a network tries to extract features that are similar across domains, have become increasingly popular [14], [21], [22].

One of the most challenging applications of domain adaptation is on the field of sensor networks. Here, we take a rather broad definition for sensor networks, defining it as a set of entities somehow related to each other and that produce a stream of data over time. Examples of sensor networks are the set of traffic cameras within a city, the set of thermometers part of a wireless monitoring system or the set of electrodes placed on the skin when performing an electrocardiography. Usually, learning models developed with data collected from a set of sensors fail to generalize well to new sensors.

1.2 Motivation

As different sensors are added to and excluded from the network, models have to take into account the fact that the sensors used to train a model are different from the ones where the model will make predictions. Since domain shifts between different sensors on a sensor network are usually substantial, it is imperative that robust domain adaptation methods are developed that take into account the constraints of the sensor network.

There is a lack of domain adaptation methods focusing on how to handle the temporal component of data. Chen et al. proposed an algorithm called Temporal Attentive Alignment for implementing DA in video datasets that explicitly attends to temporal dynamics [23] and Liu et al. proposed a spatio-temporal DA model named TrCbrBoost for classifying land use [24]. It should be noted, though, that both of these works only deal with a single-source-single-target setting, which is simpler than the multiple source case present when dealing with sensor networks.

For dealing with a multiple source setting, adversarial approaches have proven successful. Zhao et al. introduced an algorithm called multisource domain adversarial networks (MDAN) that makes use of two generative adversarial neural networks: one for discriminating between domains and another one for learning the task [14]. MDAN showed superior performance when compared to other state-of-the-art methods on the task of counting vehicles in images obtained from city cameras videos. Given the positive results obtained, and given the fact that MDAN does not consider the temporal component of the video frames, we considered worthy to investigate the adaptation of this model so that it can receive a temporal sequence as an input.

Hence, over the course of our dissertation, we explore how to adapt the MDAN model so that both of its adversarial networks are LSTM networks. We decided to go with this type of network, since it has already shown promising results when dealing with sensor networks. Zhang et al. introduced an LSTM network architecture for counting vehicles in images obtained from city cameras [10], using the same dataset in which MDAN was evaluated, and reported an improvement of the mean absolute error when compared to other state-of-the-art methods.

1.3 Objectives

By taking up this dissertation, we focus on solving the issues that prevent machine learning models from achieving a good performance when dealing with sensor networks. For that, we define two main objectives:

- Adapt the MDAN algorithm so that it considers the temporal and sequential nature of data.
- Evaluate the performance of the newly proposed algorithms on multiple datasets, so that we are able to demonstrate their efficiency.

When it comes to this report in particular, we hope that it fulfills the following goals:

- Clearly define the problem we tackle in our dissertation.

- Provide a thorough and in-depth review of the state-of-the-art methods relevant to our work.
- Explain concisely the new algorithms proposed by us.
- Present the results of the experiments we ran to test our models.

1.4 Document Structure

The remainder of this work has the following structure:

- **Chapter 2: "Deep Learning: Literature Review"** introduces the deep learning topics most relevant to our work.
- **Chapter 3: "Domain Adaptation"** provides an overview of the domain adaptation field and introduces the main literature on this topic.
- **Chapter 4: "Domain Adaptation for Counting Objects in Video Frames"** describes the models we proposed to tackle the problem of adversarial domain adaptation in sensor networks, applied to the problem of counting objects in video frames.
- **Chapter 5: "Experiments"** presents the experiments we ran to compare our models against other state-of-the-art methods and discusses the results.
- **Chapter 6: "Conclusion"** concludes the dissertation with an overview of the results we have already obtained and the possibilities for future work in this area.

Chapter 2

Deep Learning: Literature Review

This chapter provides an overview of deep learning topics that had the most impact in the work we developed in our thesis, namely: long short-term memory networks (LSTMs), convolutional neural networks (CNNs) and generative adversarial networks (GANs). We provide an overview for each of these topics and present the historical context to them. Afterwards, we discuss the deep spatio-temporal neural networks architecture, a model for counting vehicles that combines LSTMs and CNNs and that we directly use in our thesis.

2.1 Long Short-Term Memory Networks

2.1.1 Overview

Neural networks are a powerful tool that has led to tremendous advances in the field of machine learning. As the available computing power increased due to hardware advances, more complex architectures have been deployed, like convolutional neural networks (CNNs), that are able to capt underlying patterns in data, sometimes even better than humans [25]. The applications of neural networks now span diverse areas like finance [26], quantum chemistry [27] and medical diagnosis [28].

Still, despite their capabilities, traditional neural networks are unable to deal with sequential data, as they treat each sample independently. That makes them unable for dealing, for example, with speech or video. In order to address this issue, recurrent neural networks (RNN) were invented. They keep an internal state, which allows them to treat samples dependently, that is, a particular example can influence a different example that comes after in the sequence. Furthermore, their chain-like architecture allows them to model input sequences such as temporal data. Figure 2.1 shows a diagram representing the architecture of an RNN. The output $h^{(t)}$ is given by the following equation:

$$h^{(t)} = f\left(x^{(t)}, h^{(t-1)}\right) \tag{2.1}$$

where f must be a bounded function, otherwise the outputs can explode as time evolves. Typical choices are the sigmoid and the hyperbolic tangent.

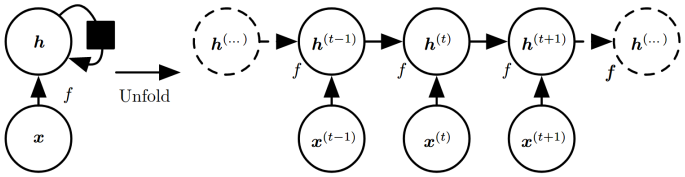

Figure 2.1: Recurrent Neural Network. Reprinted from Goodfellow et al. [1]

However, RNNs are unable to deal with long term dependencies. That is, if an example is influenced by another that comes way before in the sequence, the network will face difficulties learning that influence. The reason for that is that, as the RNN grows, its gradient either vanishes or explodes. LSTM is a type of RNN that solves this issue with its internal gates, that adaptively control how much the past should influence the future, which allows it to remember information for longer periods of time.

Figure 2.2, reprinted from Greff et al. [2], shows the LSTM internal gates: input, output and forget gate that decides which information to forget. Additionally, it contains a block input, an output activation function and peephole connections, which allow the gates access to the cell state.

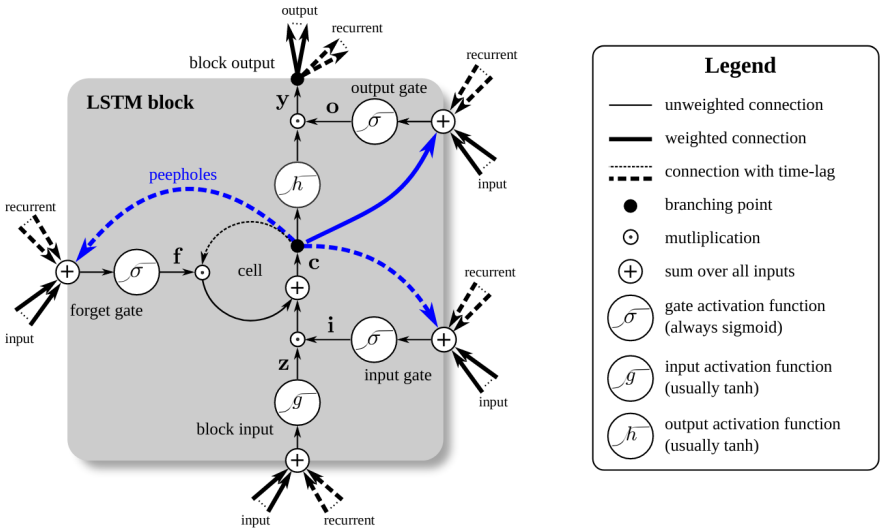

Figure 2.2: LSTM cell. Reprinted from Greff et al. [2]

In the last decade, LSTM networks have led to vast and significant practical applications such as speech recognition [29], handwriting recognition [30], traffic forecast [31] or language modelling [32].

2.1.2 History

Recurrent neural networks were first introduced in 1986 by David Rumelhart [33]. In 1997, a new type of RNN was proposed by Sepp Hochreiter and Jürgen Schmidhuber, named long short-term

memory network with the intent of addressing traditional RNNs inability to store information over long periods of time [34].

In 1999, the forget gate was introduced, allowing the network to discard irrelevant information [2] and in 2000, the actual model of LSTM became complete as peephole connections were added to the LSTM cell [2].

Over the next decade, LSTMs would revolutionize sequential data processing. From speech to handwriting, LSTMs assumed the role of state-of-the-art model when it comes to dealing with this kind of data. As of 2017, Facebook processes 4.5 billion automatic translations using LSTMs [35].

2.1.3 LSTM Applications

In this section, we discuss some LSTM applications relevant to our dissertation, due to how close they mirror the sensor networks that we intend to tackle in our work and due to the advances they brought.

2.1.3.1 Deep Recurrent Spatial-Aware Network

In 2018, Liu et al. proposed a model named Deep Recurrent Spatial-Aware Network for crowd counting [3], that is, for counting the number of people in a crowd. The model was based on LSTM networks and experiments showed a superior performance compared to other state-of-the-art methods.

The model proposed incorporates two components: the Global Feature Embedding (GFE) module that extracts features from an image and a Recurrent Spatial-Aware Refinement (RSAR) module that iteratively refines the crowd density map. This last module incorporates an important advancement, as the LSTM cell possesses a Spatial Transformer Network that allows the model to deal with different rotation variations and different scales in the same scene. Figure 2.3 is reprinted from the paper by Liu et al. and shows the architecture of the model proposed by them.

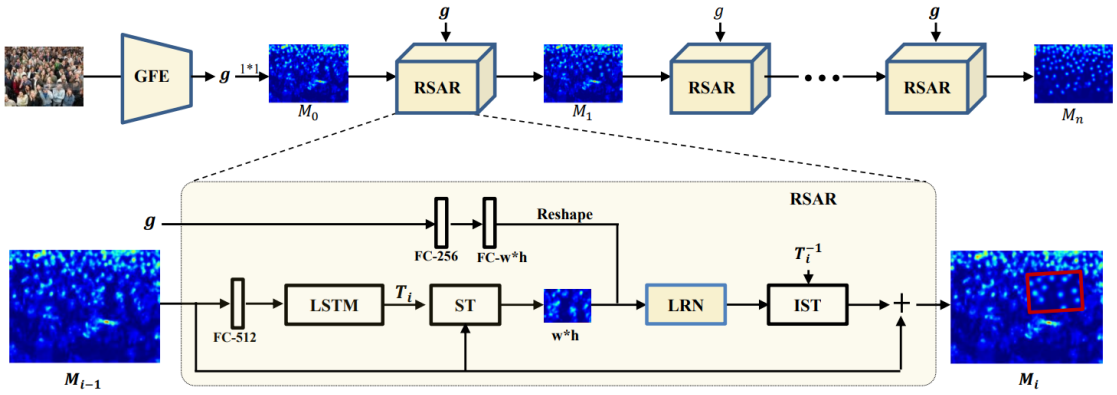

Figure 2.3: Deep Recurrent Spatial-Aware Network architecture. Reprinted from Liu et al. [3]

2.1.3.2 Spatially aware Deep Attention Models

In 2017, Varshneya et al. proposed Spatio-Temporal Attention Model, for predicting the motion patterns of humans, that is, knowing the previous trajectory of humans, predicting the location of these subjects in the next time units [4].

The solution contains a module named Spatially Static Context Network (SSCN), used to model the static spatial context around the subject of interest and based on convolutional neural networks. Afterwards, it uses a Pooling Mechanism which captures the influence that nearby objects have on the target subject. This pooling mechanism makes use of LSTMs as an attention mechanism for trajectory planning, a task that requires long-term dependency, unlike the short-term temporal dependencies in sequential data that LSTMs have traditionally been used to model. Figure 2.4 shows the Pooling Mechanism proposed by Varshneya et al.

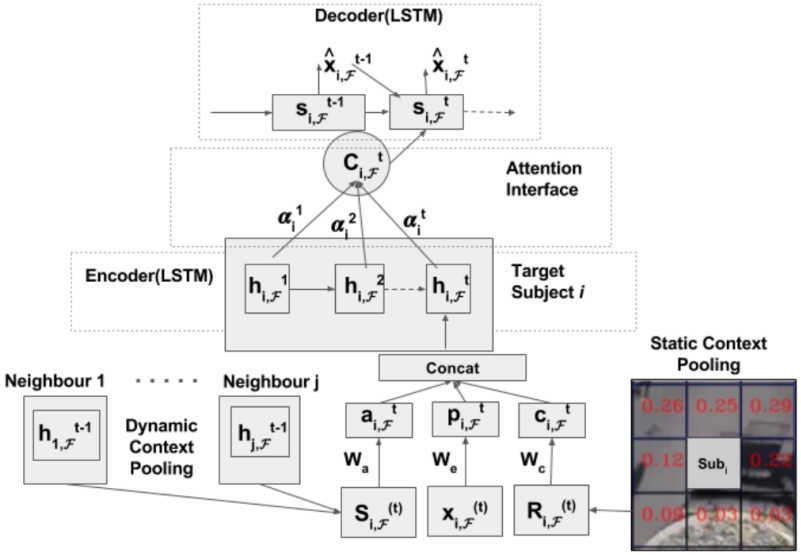

Figure 2.4: Pooling Mechanism architecture. Reprinted from Varshneya et al. [4]

2.2 Convolutional Neural Networks

2.2.1 Overview

Convolutional neural networks (CNNs) are a type of neural network that can receive images as the input, being able to learn patterns and differentiating elements of those same images. The preprocessing required for a CNN is significantly lower than that of traditional image classification algorithms, as the CNNs learn filters that used to be hand-crafted in traditional algorithms. When large annotated datasets are available, the performance of CNNs is also often considerably superior when compared to any model based on hand-crafted features.

The architecture of CNNs was inspired by biological processes mimicking the organization of neurons in the visual cortex of the animal brain. Individual neurons respond only to a particular

region of the visual field, named receptive field. The receptive fields overlap in order to cover the entirety of the visual field.

Like standard neural networks, convolutional neural networks are feedforward networks, containing an input and output layer and multiple hidden layers. The hidden layers typically consist of traditional fully connected layers, but also of convolutional and pooling layers, designed specifically for CNNs. A deep neural network is generally called a CNN whenever it has one or more convolutional layers.

2.2.1.1 Convolutional Layer

The convolution operation is the mathematical basis of the convolutional layer. Next, we present its definition.

Definition 2.2.1. Convolution Operation
Consider a matrix I with shape (c_I, h_I, w_I) where c_I is the number of channels in I, h_I is the height and w_I the width, and a kernel K with shape (c_K, c_I, h_K, w_K), where c_K is the number of output channels in K, h_K is the height and w_K is the width. The result of the convolution of I by K, $I * K$ is defined as:

$$(I * K)[l,m,n] = \sum_{c=0}^{c_I-1} \sum_{i=0}^{h_K-1} \sum_{j=0}^{w_K-1} K[l,c,i,j] \times I[c, m+i, n+j] \tag{2.2}$$

so that $I * K$ has shape $(c_K, h_I - h_K + 1, w_I - w_K + 1)$.

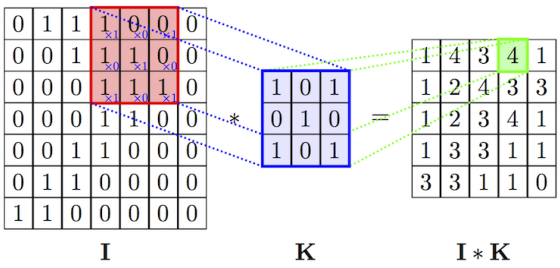

Figure 2.5: Example of convolution operation. In this case, K and I have only one channel. Reprinted from [5]

The convolutional layer is a feature extractor that maps the image received as input to a new feature space. Each neuron looks at a particular region of the input, so that the layer is able to retrieve local properties of the image. Inputs are convolved with a kernel filter, generating a new feature map and the results are then sent through a nonlinear activation function like ReLU.

More formally, if the input of this layer is a tensor X with shape *(batch size, input channels, image height, image width)* and the kernel filter K has shape *(output channels, kernel height, kernel width)*, this generates an output Y on a new feature map, such that $Y = a(K * X)$ and Y has shape *(batch size, output channels, feature map height, feature map width)*, where a is an activation function.

There are two particular properties that make convolutional layers specially suited for image data: translation equivariance and sparse interactions. Translation equivariance refers to the fact that a translation of the input produces an identical translation in the output. Sparse interactions are a consequence of the reduced size of the convolutional kernels when compared to the weight matrices of fully convolutional layers. Thus, in a convolutional layer, the value of an output neuron is usually determined by only a few input neurons. In a fully connected layer this is not the case, since, by definition, all input neurons are connected to all output neurons.

2.2.1.2 Pooling Layer

Pooling layers are typically placed between convolutional layers and their purpose is to reduce the spatial size of the image in order to reduce the computation time and to control for overfitting. They combine the outputs of neuron clusters at one layer into one single neuron at the next layer. Common functions used for combining outputs of a cluster are *max*, *min* and *average*. Figure 2.6 shows a comparison between MaxPool and AveragePool.

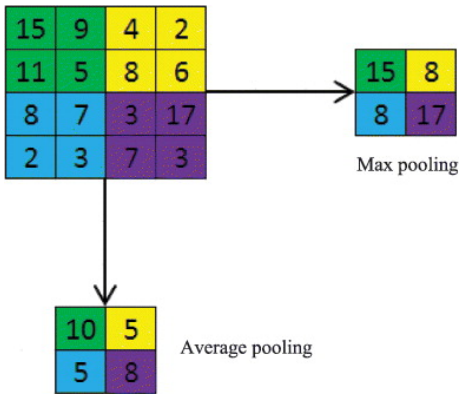

Figure 2.6: Comparison between MaxPool and AveragePool. Reprinted from [6]

2.2.1.3 Atrous Convolutional layer

Atrous Convolutional layers are an extension of convolutional layers, introducing a new parameter called dilation rate, that defines the spacing between the cells of the kernel. For example, a kernel with dimensions 4×4 and a dilation rate of 2, would have the same field of view as a kernel with dimensions 7×7. An atrous convolution layer is able to amplify the field of view compared to a regular convolutional layer without increasing the number of parameters. Figure 2.7 shows an example of an atrous convolution layer with different dilation rates.

2.2.2 History

In 1959, neurophysiologists David Hubel and Torsten Wiesel showed that the animal visual cortex is composed of simple and complex cells. A simple cell responds to edges with a certain orientation, whereas a complex cell also responds to edges with a particular orientation with the

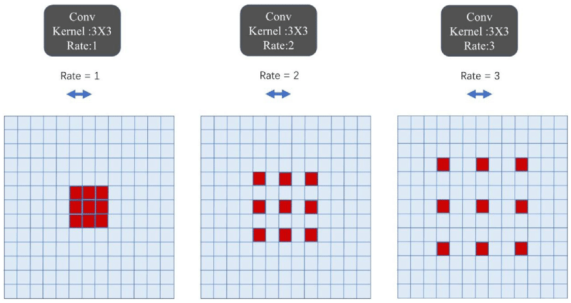

Figure 2.7: Example Atrous convolution with kernel size 3×3 and with different dilation rates. Reprinted from [7]

difference that the location of these edges can be changed around the image and the cell would still respond. In 1962, the same scientists proposed that complex cells are actually a collection of simple cells, such that each simple cell is activated by a particular region of the image, with the parent complex cell then summing the outputs of the simple cells. Their work introduced the concept of receptive field, the region of the visual field where a particular neuron is activated, that would later become the basis of convolutional neural networks.

In 1980, inspired by the work of Hubel and Wiesel, Kunihiko Fukushima proposed the model Neocognitron [36], a neural network with convolutional layers, where the image is convoluted with a kernel and downsampling layers, where the spatial size of an image is reduced. In 1998, inspired by the Neocognitron, Yann Le Cunn. et al. proposed a CNN for handwriting recognition [37], a model which progressively aggregates simple features into more complex ones and that can be described as the first work on modern CNNs.

CNNs surged in popularity in the 2000s, due to the availability of GPUs that decreased their training time by up to 20 times when compared to CPUs. The first implementation of CNNs on GPUs was proposed by K. Chellapilla et al. in 2006 [38]. In 2011, Ciresan et al. proposed a method to train CNNs with many layers on GPUs that was able to obtain an acceleration factor of 60 [39]. This method would allow for the record-breaking performance of CNNs in the next decade. In 2012, a CNN named AlexNet achieved state-of-the-art performance in the ImageNet challenge. The paper by Alex Krizhevsky et al. describing its architecture has since been cited more than 35000 times [40]. Convolutional neural networks have, in the last years, often achieved superhuman performance in some tasks, from facial recognition [41] to identifying skin cancer lesions [42].

2.3 GANs

Generative Adversarial Networks (GANs) are a novel concept, introduced in 2014 by Goodfellow et al. [43], but having already led to tremendous progress in multiple areas of machine learning. As Yann LeCun described them: "the most interesting idea in the last 10 years in Machine Learning".

A GAN is a framework that tries to generate new data with the same distribution as the data that was fed into it as input. It consists of two models: a generator that generates data receiving

random noise as an input and a discriminator that tries to distinguish between real data and data generated by the generator, so that they both play a two player minimax with the following value function:

$$\min_G \max_D \left(E_{x \sim p_{data}(x)} \left[\log D(x) \right] + E_{z \sim p_{data}(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right) \tag{2.3}$$

where $p_{data}(x)$ represents the distribution of real data and $p_z(z)$ represents the distribution of noise given as an input to the generator. $D(x)$ represents the probability of data x being real, whereas $G(z)$ represents the generator output when z is given to it as an input.

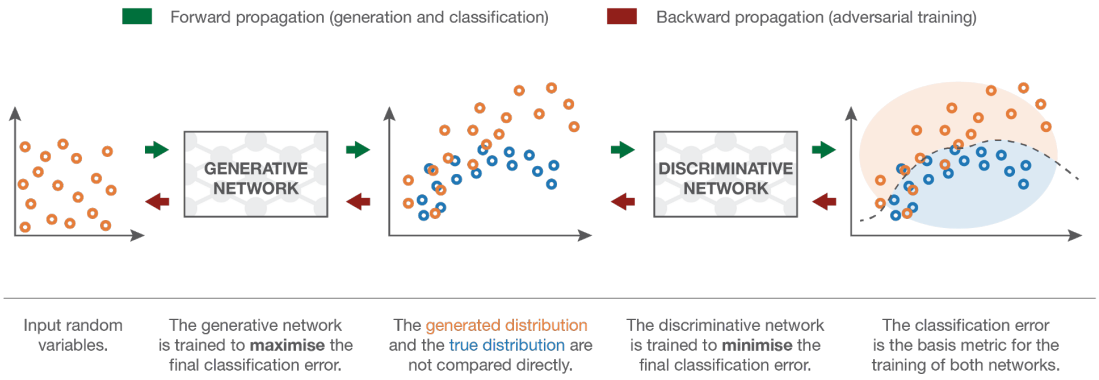

Figure 2.8: Generative Adversarial Networks representation. Reprinted from [8]

Despite being a recent idea, GANs have already led to tremendous advances in multiple areas. They have been used to model the distribution of dark matter and predict its gravitational lensing [44], they have been used used in fashion photo-shoots to create photographs with imaginary models, dispensing the need for hiring expensive professionals [45] and they have been used to generate images from textual descriptions of objects like flowers and birds [46].

As mentioned before, GANs have also inspired recent and significant advances in the field of domain adaptation, due to the introduction of adversarial domain adaptation, where a network minimizes a loss related to the discrimination between different domains and regression or classification loss related to a particular task. This is the application of GANs that interests us the most in our thesis.

GANs have also brought some concerns, due to the possibility of its use by actors with a malicious intent. One of the sinister applications of GANs that has caught the most attention is a deep learning technique named DeepFakes that seeks to create video by replacing the face of a person with the face of another [9]. As this technique becomes more robust, we see increasing discussion on how to deal with its harmful uses like fake pornography, fake news and identity theft. Figure 2.9 shows examples of celebrities that had their expressions swapped with another through the use of DeepFakes.

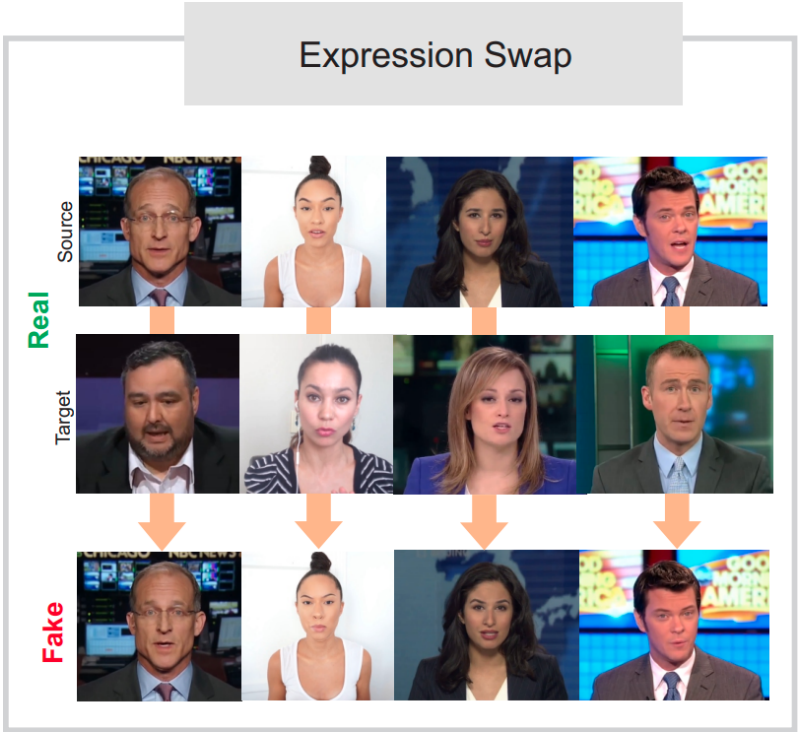

Figure 2.9: Expression swap using DeepFakes. Reprinted from [9]

2.4 FCN-rLSTM: Deep Spatio-Temporal Neural Networks for Vehicle Counting

In 2017, Zhang et al. introduced an LSTM network architecture for counting vehicles in images obtained from city cameras, named FCN-rLSTM [10]. The model allows for the consideration of the temporal component of data. That is, in order to determine the number of vehicles in a frame, it also uses information extracted from previous frames.

According to Zhang et al., methods for counting vehicles in videos can be divided into five categories:

- **Frame differencing** methods count vehicles based on the difference between sequential frames [47];
- **Detection based methods** detect individual vehicles and count their total number [48];
- **Motion based methods** count vehicles by tracking them [49], [50];
- **Density estimation based methods** map the image features into vehicles densities, casting the counting problem as an estimation of an image density whose integral gives the count of vehicles within that image [51];
- **Deep learning based counting methods** are the most recent and use deep learning techniques like CNNs and ridge regressors to output the number of vehicles in the image without explicitly looking for individual vehicles or estimating density maps [52], [53].

Issues with the quality of the collected data can make vehicle counting a difficult problem, namely low frame rate, low resolution, vehicle occlusion and different vehicle scales, particularly noticeable when the camera is too close to the road. Thus, the nature of the frames in each dataset has to be taken into account when determining which category of vehicle counting methods to use. The FCN-rLSTM is a density based estimation method, being able to deal well with a low frame rate, a low resolution and vehicle occlusion but having difficulty in accounting for different vehicle scales.

The proposed network uses a Convolutional Neural Network to obtain a global estimate of the number of vehicles and a chain of LSTMs for a residual estimation of this same number, that is going to be added to the global estimate. Figure 2.10 is reprinted from the article published by Zhang et al. and shows the proposed architecture.

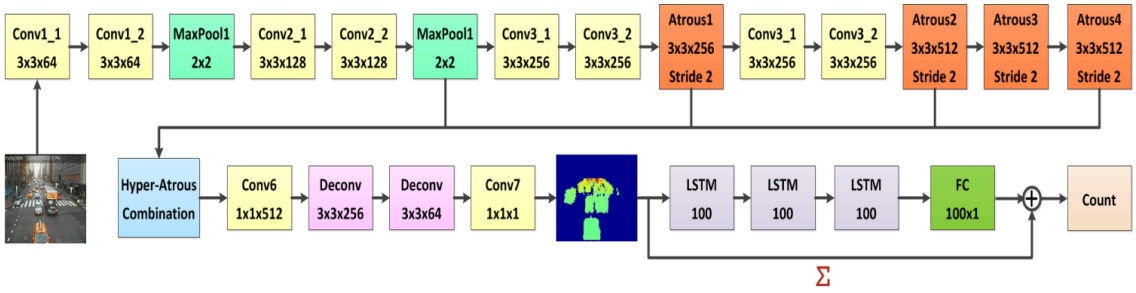

Figure 2.10: Network architecture. Reprinted from Zhang et al. [10]

The network receives as ground truth the density map and the global count for each frame. In each density map, a Gaussian kernel is placed around the center of each car, so that the total sum of the kernel is 1 and so that its standard deviation is proportional to length of the car. For the computation of the loss, a loss L_C is calculated as being the mean squared error of the frame vehicle counts predicted by the network and a loss L_D is calculated as being the mean squared error of the frame density maps predicted by the network. The global loss is then:

$$L_D + \lambda L_C \tag{2.4}$$

where $\lambda > 0$ is a hyperparameter that controls the weight to be given to the count loss relative to the density loss.

The Hyper-Atrous combination that merges the outputs of different atrous convolutional layers was introduced for dealing with reduced feature resolution caused by the multiple MaxPool layers. The convolutional layer with kernel 1×1 that follows it is then used for feature re-weighting and for distinguishing better foreground from background, without the need to carry out the time-consuming task of foreground segmentation. This re-weighted feature map then goes through two deconvolution layers and, after that, is passed as an input to a convolutional layer with kernel 1×1 that acts as a regressor to map the features into a density map.

The FCN-rLSTM was the first architecture to introduce a residual connection for object counting, where an LSTM estimates the residual between the sum of the density map and the final pre-

dicted count value. The setting with residual connection is able to significantly reduce the training time, while also increasing the counting accuracy. Figure 2.11 shows a comparison between a directly connected model and a residual connected one.

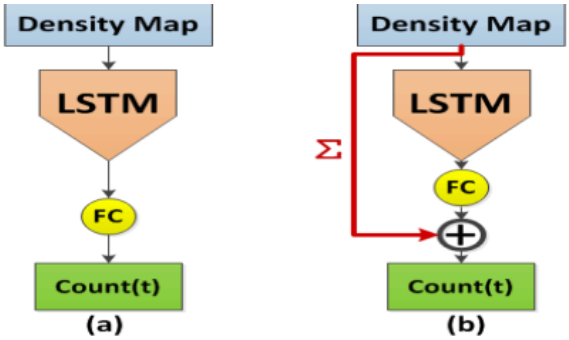

Figure 2.11: Comparison between a) direct connection and b) residual connection. Reprinted from Zhang et al. [10]

We adopt the FCN-rLSTM architecture in the models we propose in this dissertation, due to the good results it obtained in the experiments when compared to other methods, due to the fact that it considers the temporal component of data and because we believe the convolutional neural network (CNN) architecture presented is ideal for counting objects in video frames. This architecture provides a fast training time and is able to effectively deal with multiple scenarios like weather conditions or a large perspective.

Chapter 3

Domain Adaptation: Literature Review

This chapter provides an overview of the field of domain adaptation. It starts with an introduction that defines the term and provides a context to it. It is followed by a section presenting a historical context to the area. Afterwards, the traditional domain adaptation methods are explained and discussed. Lastly, we analyse more in-depth some adversarial domain adaptation algorithms that have inspired our work.

3.1 Introduction

Traditional machine learning algorithms assume that the distribution of data in the training and testing datasets is the same. Unfortunately, it is often not possible to annotate training data from the same source the testing data comes from. In that case, the performance of the algorithm will be degraded. Domain adaptation (DA) seeks to address that by enabling the transference of learning from a model in a source data distribution to a different target data distribution.

Domain adaptation mimics the biological learning process in humans. A polyglot that knows multiple languages will have an easier time learning a new language than someone that can only speak his or her native tongue [54]. That is because our brains are able to adapt and transfer the knowledge we acquired when learning a language to another unrelated one.

Regarding a real application of domain adaptation in machine learning, consider the task of determining the word a photograph of a sign language gesture corresponds to [22]. It would be hard to collect images from a significant number of people, due to the difficulty of finding a high number of volunteers with knowledge of sign language and the logistics of having many participants in our study. As such, it would not be possible to train our model with data that is truly representative of the general human population. We can then use domain adaptation in order to adapt the data collected to a more generic population. The basic idea behind it would be to discard signer-specific information while identifying and preserving information about the signs that is common to every signer.

Domain adaptation algorithms can be classified according to the labels of data:

- **supervised domain adaptation**, where all of the examples in both the source and the target examples are labeled.
- **semi-supervised domain adaptation**, where a set of examples in both the source and target domain are not labeled.
- **unsupervised domain adaptation**, where only a set of examples in the source domain are labeled.

Unsupervised domain adaptation has grown in importance with the advent of Big Data, due to the difficulty of manually annotating large-scale datasets, which demands a method capable of transferring knowledge from a labeled source domain to an unlabeled target domain. Unsupervised domain adaptation is able to perform this task by identifying domain-invariant features and develop machine learning models with those features instead of domain-specific ones.

Domain adaptation is currently an established field that has led to multiple practical applications. In the biomedical world, it has been used for cancer subtype discovery from genome-scale profiling [55], for the segmentation of radiological images [56] or for medical imaging classification [57]. When it comes to Natural Language Processing (NLP), its influence is also significant as it has been used for adapting a model from a language to another [58], for adapting a model trained with text related to a certain topic to text related to a different topic [59] or for semantic role labelling of clinical text [60]. Other applications of DA also include video classification [61] and counting vehicles from city cameras [14].

3.2 History

Domain adaptation is actually a sub-field of transfer learning (TL). In transfer learning, we do not seek to simply adapt knowledge from one domain to another, but also to adapt knowledge from one task to another. That means domain adaptation is a sub-case of transfer learning where the task to be performed in the source domain is the same as the one to be performed in the target domain.

In 1980, Tom Mitchell explored how the use of bias can help generalize a task [62]. In 1986, Paul Utgoff built upon the work of Mitchell and proposed a framework for shifting bias over the course of inductive concept learning training [63]. In 1993, Lorien Pratt published a paper introducing the discriminality-based transfer algorithm (DBT) for the transference of learning between two neural networks [64]. In 1997, the *Machine Learning* journal published an issue entirely devoted to transfer learning, indicating the rising importance of the field [65], and in 1998 the formal definition of multi-task learning as we know it today was introduced [66].

In the following decade, works that specifically addressed the problem of domain adaptation started to appear, first with reweighting algorithms [67], [68] and then with feature-based algorithms [69], [70], where the domain adaptation method tries to identify a set of common features to both the source and the target domains.

More recently, with the surge of deep learning, new domain adaptation methods that leverage the power of this field have appeared [71], [72]. These new methods are better able to tackle the challenges presented by the massive amounts of unlabeled data that came as a consequence of the advent of Big Data.

3.3 Notation

The following notation will be used throughout this chapter:

- χ is the input space
- γ is the output space
- X is the input variable.
- Y is the output variable.
- $P(X)$ and $P(Y)$ represent the distribution of X and Y, respectively.
- $P(X,Y)$ represents the joint distribution of X and Y.
- $P_s(X)$ and $P_s(Y)$ are the distributions of X and Y in the source domain, respectively.
- $P_t(X)$ and $P_t(Y)$ are the distributions of X and Y in the target domain, respectively.
- D_s is the joint distribution of X and Y in the source domain and D_t is the joint distribution of X and Y in the target domain. In the case of multisource domain adaptation, $\{D_{s_1}, D_{s_2}...D_{s_k}\}$ are the joint distributions of X and Y in the source domains and $\{D_{t_1}, D_{t_2}...D_{t_k}\}$ are the joint distributions of X and Y in the target domains.
- Given a classifier $\eta : \chi \rightarrow \gamma$ in a particular domain D, we define its risk $\varepsilon_D(\eta)$ as:

$$P_{(x,y)\sim D}(\eta(x) \neq y) \tag{3.1}$$

- Given the same classifier $\eta : \chi \rightarrow \gamma$ in a particular domain D and a set of n examples $(x_1,y_1),(x_2,y_2)...(x_n,y_n)$, we define its empirical risk $\widehat{\varepsilon}_D(\eta)$ as:

$$\frac{1}{n}\sum_{i=1}^{n} I(\eta(x_i) \neq y_i) \tag{3.2}$$

  where $I(c)$ is the predicate function, being 1 if the condition c is true and 0 if it is false.

3.4 Traditional Domain Adaptation Methods

This section presents an overview of traditional domain methods. The methods were chosen taking into account their importance in the literature by the introduction of new concepts in the area and their popularity.

Domain adaptation methods can be classified according to some properties they assume about data:

- **target shift**, where the marginal distribution $P(Y)$ of labels changes, but the conditional $P(X|Y)$ distribution of features given labels is the same across domains [73], [74], [75];
- **conditional shift**, where $P(Y)$ is the same and $P(X|Y)$ changes [76], [77];
- **covariate shift**, where the marginal distribution $P(X)$ of features changes but the conditional $P(Y|X)$ distribution of labels given features is the same [78], [79];
- **generalized target shift** which combines target shift and conditional shift by allowing $P(Y)$ to change and $P(X|Y)$ to also change with some restrictions [80].

In the next section we present examples for some of these categories.

3.4.1 Reweighting algorithms

Reweighting algorithms seek to reweight the source distribution so that it looks like the target one. Let us first review the theoretical justification for instance weighting introduced by Jiang [67].

Let Θ be a model family from which we intend to select an optimal model θ^*. Let $l(x,y,\theta)$ be a loss function. For a certain distribution $P(X,Y)$, we intend to select the model θ that minimizes:

$$\sum_{(x,y)\in\chi\times\gamma} P(x,y)l(x,y,\theta) \tag{3.3}$$

In the setting of domain adaptation, we intend to minimize this quantity for the target domain, so that:

$$\begin{aligned}\theta_t^* &= \underset{\theta\in\Theta}{\operatorname{argmin}} \sum_{(x,y)\in\chi\times\gamma} P_t(x,y)l(x,y,\theta) \\ &= \underset{\theta\in\Theta}{\operatorname{argmin}} \sum_{(x,y)\in\chi\times\gamma} \frac{P_t(x,y)}{P_s(x,y)} P_s(x,y)l(x,y,\theta).\end{aligned}$$

Using a given set of instances from the source domain, $\{(x_i^s,y_i^s)\}_{i=1}^{N_s}$, the expectation above is approximated by its empirical version:

$$\theta_t^* \approx \underset{\theta\in\Theta}{\operatorname{argmin}} \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{P_t(x_i^s,y_i^s)}{P_s(x_i^s,y_i^s)} l(x_i^s,y_i^s,\theta). \tag{3.4}$$

As such, the optimal weight for instance (x_i^s, y_i^s) is $\frac{P_t(x_i^s, y_i^s)}{P_s(x_i^s, y_i^s)}$. Since the true distributions P_s and P_t are unknown, these weights are approximated heuristically, using a (small) set of labeled instances from the target domain.

3.4.1.1 Class imbalance

Some works have focused on the issue of class imbalance between the source and target domains [81], [82], [83]. Under the target shift setting, it is assumed that $P_s(X|Y=y) = P_t(X|Y=y)$, for all $y \in \gamma$, but $P_s(Y) \neq P_t(Y)$. The weight for instance (x,y) is then:

$$\frac{P_t(x,y)}{P_s(x,y)} = \frac{P_t(y)}{P_s(y)} \frac{P_t(x|y)}{P_s(x|y)} = \frac{P_t(y)}{P_s(y)} \tag{3.5}$$

These weights can then be determined empirically, given that there is a significant number of instances belonging to each class in Y.

3.4.1.2 Covariate Shift

In covariate shift [84], it is assumed that $P_s(Y|X=x) = P_t(Y|X=x)$, for all x in the input space, but $P_s(X) \neq P_t(X)$. Under these assumptions, the weight for instance (x,y) can be calculated as:

$$\frac{P_t(x,y)}{P_s(x,y)} = \frac{P_t(x)}{P_s(x)} \frac{P_t(y|x)}{P_s(y|x)} = \frac{P_t(x)}{P_s(x)} \tag{3.6}$$

A major difficulty arises when calculating $\frac{P_t(x)}{P_s(x)}$. Shimodaira et al. proposed to use non-parametric kernel estimation [84]. Other works explore how to turn this calculation into a problem of predicting whether an example is from the source domain or the target domain [85], [86].

3.4.2 Iterative algorithms

Iterative algorithms [19], [20] are used for labeling unlabeled data in the target domain. The method is as follows:

1. Train a model with the labeled samples
2. Classify some unlabeled samples
3. Train a new model with the now labeled samples
4. Classify some other unlabeled samples

This process is repeated several times until all samples are labeled.

3.4.3 Hierarchical Bayesian

Finkel and Manning published a work that proposes a method named hierarchical Bayesian domain adaptation [58], which makes use of a hierarchical Bayesian prior. Each domain has its

own domain-specific parameter (weight) for each feature, which the model links via a hierarchical Bayesian global prior, encouraging features to have similar weights across domains. It should be noted, though, that unlike other methods, the Hierarchical Bayesian does not treat one domain as the source and another as the target. Instead, it tries to assign parameter values for each feature, linking information from every domain.

On a formal setting, consider that we have a domain D with a set of parameters θ, so that the parameter for a particular feature is θ_j. Assume that the parameter for each feature has a zero-mean Gaussian prior over it, with its standard deviation being σ. The log-likehood function for domain D is $L(D,\sigma)$. The set of parameters chosen $\widehat{\theta}$ is given by the following formula:

$$\widehat{\theta} = \underset{\theta}{\operatorname{argmax}}\, L(D,\theta) - \sum_j \frac{\theta_j^2}{2\sigma^2} \tag{3.7}$$

We can extend this single-domain formula to a multi-domain scenario. In this case, there are k domains $\{D_1, D_2 ... D_k\}$, each one with a set of parameters θ_i, so that the parameter for a particular feature is θ_{ij} and its standard deviation θ_i. Domains are connected with each other through a tree-like structure, that has domains D_i as leaves. The root of this tree and parent of every domain is an auxiliary imaginary domain, whose parameters are denoted by θ_*.

The formula for deciding the parameters of every domain on a multi-domain scenario is:

$$\widehat{\theta} = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{k} \left(L(D_i,\theta) - \sum_j \frac{(\theta_{ij} - \theta_{*j})^2}{2\sigma_i^2} \right) - \sum_j \frac{\theta_{*j}^2}{2\sigma_*^2} \tag{3.8}$$

3.4.4 Divergence-based Domain Adaptation

Divergence-based domain adaptation methods find a domain-invariant feature representation where a divergence criterion between the source and target distributions is minimized. The classifier can then be trained on that representation and have a similar performance on both domains.

There are several divergence criteria that can be used for domain adaptation. In this section we present two: maximum mean discrepancy (MMD) and correlation alignment (CORAL).

3.4.4.1 Maximum Mean Discrepancy

Maximum Mean Discrepancy is a measure for the distance of two distributions on a Reproducing Kernel Hilbert Space.

Definition 3.4.1. Maximum Mean Discrepancy
Consider distributions P and Q over a set χ. Consider also a feature map $\varphi : \chi \to H$, where H is a Reproducing Kernel Hilbert Space. Then, MMD is defined as:

$$MMD(P,Q) = ||\mathbb{E}_{X_1 \sim P}[\varphi(X_1)] - \mathbb{E}_{X_2 \sim Q}[\varphi(X_2)]||_H \tag{3.9}$$

Rozantsev et al. published a work that leverages this concept to implement domain adaptation [11]. It introduces a deep learning based two-stream architecture that can be seen in figure 3.1.

The first stream handles the source domain, whereas the second one handles the target domain. The weights in both the source and the target domain are allowed to differ somewhat but not considerably, due to regularization. MMD was used as loss.

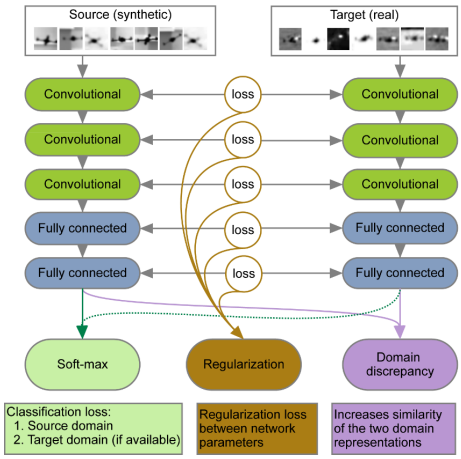

Figure 3.1: Two-stream architecture for domain adaptation using MMD. Reprinted from Rozantsev et al. [11]

3.4.4.2 Correlation Alignment

A paper by Sun et al. introduced the algorithm Correlation Alignment (CORAL) [12], a domain adaption method that also uses a distance-measure between distributions as a divergence criterion. CORAL tries to minimize the distance between source and target distributions, by aligning the second order statistics of both distributions without requiring any labels.

More formally, if C_S and C_T are the source and target covariance matrices, respectively, then the CORAL loss that the algorithm tries to minimize is defined as:

$$\min_{A} ||A^T C_S A - C_T||_F \tag{3.10}$$

where $||.||_F$ is the Frobenius norm.

Figure 3.2 shows an example of a two-stream architecture based on deep learning that leverages the CORAL algorithm for domain adaptation.

3.5 Adversarial Domain Adaptation

In an adversarial domain adaptation setting, there is a network named feature extractor that maps the inputs received into a new feature representation, a network that tries to execute the learning task, receiving samples in the new feature representation and another network that tries to discriminate between source and target features. As the classification or regression loss and the domain discrimination loss are minimized and the feature extractor is trained to fool the discriminator, the

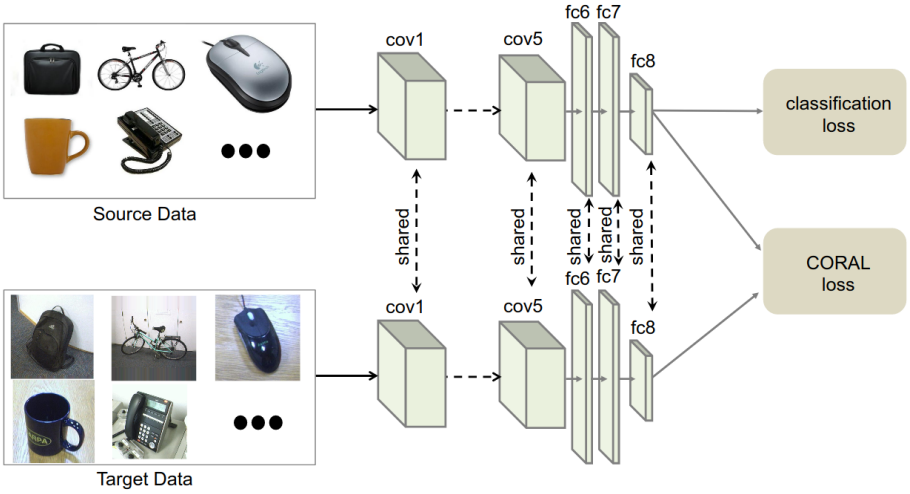

Figure 3.2: Two-stream architecture for domain adaptation using CORAL. Reprinted from Sun et al. [12]

model will generate a domain-invariant feature representation, that is still discriminative enough for executing the learning task with success.

3.5.1 Single Source Adversarial Domain Adaptation

We first introduce some definitions and theorems that explain the theoretical model of domain adaptation in a single source setting.

Ben-David et al. introduced a measure for the discrepancy of two distributions, named H-divergence [70].

Definition 3.5.1. H-divergence (Ben-David et al.)
A binary classifier η can be defined as a hypothesis $\eta : \chi \to \{0,1\}$. Let H be a hypothesis class, that is, a collection of hypotheses, and let D_s and D_t be two domains over χ. The H-divergence between D_s and D_t is:

$$d_H(D_s, D_t) = 2 \sup_{\eta \in H} |P_{x \sim D_s}(\eta(x) = 1) - P_{x \sim D_t}(\eta(x) = 1)| \tag{3.11}$$

The empirical H-divergence between two sets of samples S and T is denoted as $\widehat{d_H}(S,T)$:

$$\widehat{d_H}(S,T) = 2 \sup_{\eta \in H} |P_{x \in S}(\eta(x) = 1) - P_{x \in T}(\eta(x) = 1)| \tag{3.12}$$

Vladimir Vapnik and Alexey Chervonenkis introduced a measure for the capacity of a classifier to learn a set of data, named Vapnik–Chervonenkis dimension or, as it is commonly known, VC-dimension [87].

Definition 3.5.2. VC-dimension (Vapnik et al.)
A classifier η with parameter θ is said to shatter a set of data points $(x_1, x_2, ..., x_n)$ if, for all

assignments of labels to those points, there is a θ such that the classifier η makes no errors when evaluating that set of data points. The VC-dimension of η is the maximum number of points that can be arranged so that η shatters them.

3.5.1.1 A Generalization Bound on the Target Risk

Ben-David et al. [70] proved a bound for the target risk of a classifier η on a hypothesis class H, dependent on the source risk of that classifier, on the empirical H-divergence of H, $\widehat{d_H}$, and on the *VC dimension* of H

Theorem 3.5.1. *Given a hypothesis class H of VC dimension d, a labeled set of n samples from the source domain and an unlabeled set of n samples from the target domain, then, with probability at least* $1-\delta$ *over the choice of samples* $S \sim (D_s)^n$ *and* $T \sim (D_t)^n$*, for every* $\eta \in H$*:*

$$\varepsilon_{D_t}(\eta) \leq \widehat{\varepsilon}_S(\eta) + \sqrt{\frac{4}{n}(d\log\frac{2en}{d} + \log\frac{4}{\delta})} + \widehat{d_H}(S,T) + 4\sqrt{\frac{1}{n}(d\log\frac{2n}{d} + \log\frac{4}{\delta})} + \beta \tag{3.13}$$

with $\beta \geq \inf_{\eta* \in H}[\varepsilon_{D_s}(\eta*) + \varepsilon_{D_t}(\eta*)]$

This theorem implies that the target risk is minimized when β is low, that is, when there is a classifier that can achieve a low risk on both domains. Additionally, both the source risk and the empirical H-divergence should be minimized. A way to achieve that would be to find a classifier operating on top of a learned feature space where the source and target distributions coincide. This is the idea exploited by algorithms that aim to learn domain-invariant representations.

3.5.1.2 Domain-Adversarial Neural Networks (DANN)

A work by Ganin et al. was the first to use adversarial networks for finding a domain-invariant representation, with the introduction of an algorithm named Domain-Adversarial Neural Networks (DANN) [13]. It is inspired by the idea of GANs, consisting of three models: one that extracts features from the data in the source and target domains, a domain discriminator that tries to distinguish between domains, after the samples had been converted to the new feature space, and a model that executes the learning task. The feature extractor is trained to fool the discriminator, while producing discriminative features for the desired learning task. This way, a domain-invariant feature space will be generated that could be used to train the model, while also being discriminative for the main learning task. Figure 3.3 shows the model proposed by Ganin et al.

More formally, the model can be summarized as follows. Assume we have annotated samples $\{(x_i^s, y_i^s)\}^n$ from source domain D_s and unlabeled samples $\{x_i^t\}^{n'}$ from target domain D_t. Let $G_f(\cdot,\theta_f)$, $G_y(\cdot,\theta_y)$ and $G_d(\cdot,\theta_d)$ denote the feature extractor, the task classifier and the domain discriminator, respectively.

The prediction loss for a given sample x_i is defined as $L_y(\theta_f,\theta_y)(x_i) = L_y(G_y(G_f(x_i,\theta_f),\theta_y),y_i)$ and the domain loss for a given sample x_i is defined as $L_d(\theta_f,\theta_d)(x_i) = L_d(G_d(G_f(x_i,\theta_f),\theta_d),d(x_i))$,

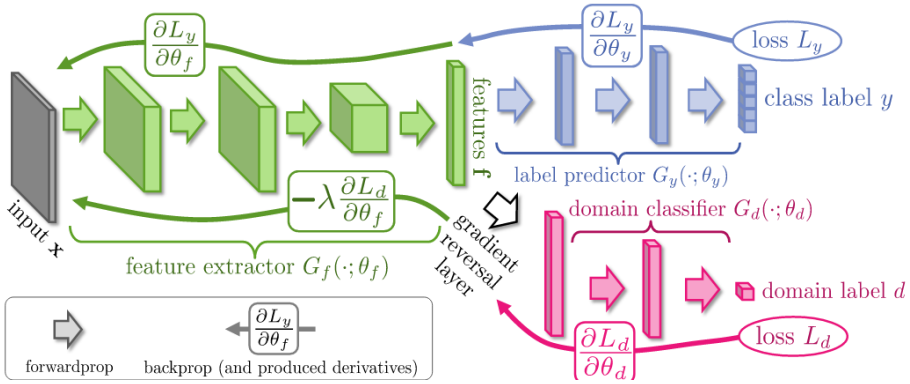

Figure 3.3: Adversarial Domain Adaptation Architecture. Reprinted from Ganin et al. [13]

where L_y and L_d are the standard classification losses for the learning task and domain discrimination task, respectively, and $d(x_i)$ is 1 if x_i is a source example and 0 if it is a target one. $\mu > 0$ is the hyperparameter that controls the weight to be given to the domain loss.

Training the model consists in optimizing:

$$E(\theta_f, \theta_y, \theta_d) = \frac{1}{n}\sum_{i=1}^{n} L_y(\theta_f, \theta_y)(x_i^s) - \mu\left(\frac{1}{n}\sum_{i=1}^{n} L_d(\theta_f, \theta_d)(x_i^s) + \frac{1}{n'}\sum_{i=1}^{n'} L_d(\theta_f, \theta_d)(x_i^t)\right) \tag{3.14}$$

by finding the saddle point $(\widehat{\theta}_f, \widehat{\theta}_y, \widehat{\theta}_d)$ such that:

$$(\widehat{\theta}_f, \widehat{\theta}_y) = \underset{\theta_f, \theta_y}{\operatorname{argmin}}\, E(\theta_f, \theta_y, \widehat{\theta}_d) \tag{3.15}$$

$$\widehat{\theta}_d = \underset{\theta_d}{\operatorname{argmax}}\, E(\widehat{\theta}_f, \widehat{\theta}_y, \theta_d) \tag{3.16}$$

3.5.1.3 Gradient Reversal Layer

Given a layer in a neural network with an activation function $a(x)$, let us denote the manually defined derivative to be used in backpropagation as:

$$\frac{\tilde{\partial} a}{\partial x} \tag{3.17}$$

Note that even though $\frac{\tilde{\partial} a}{\partial x} = \frac{\partial a}{\partial x}$ by default in most deep learning libraries, and the derivative of an activation function in a neural network model is defined as equal to its real mathematical derivative most of the time, that is not always the case.

The chain rule is also applied to manually defined derivatives, so that:

$$\frac{\tilde{\partial} z}{\partial x} = \frac{\tilde{\partial} z}{\partial y}\frac{\tilde{\partial} y}{\partial x} \tag{3.18}$$

Most adversarial domain adaptation frameworks make use of a **gradient reversal layer**. That is a layer that acts as the identity function during the forward pass but that multiplies the gradient by -1 during backpropagation. More formally, if $g(x)$ is the gradient reversal layer function, then:

$$g(x) = x \tag{3.19}$$

$$\frac{\tilde{\partial} g}{\partial x} = -1 \tag{3.20}$$

Despite its simple definition, this layer simplifies considerably the adversarial domain adaptation process. In this section, we explain the theoretical basis for the usage of the gradient reversal layer.

In order to solve the optimization problem described in equations (3.14), (3.15) and (3.16) using gradient-based (stochastic) optimization, we must choose an initial point $(\theta_f^{(0)}, \theta_y^{(0)}, \theta_d^{(0)})$ which is then updated iteratively according to the gradient descent/ascent update formulas:

$$\theta_f^{(j+1)} = \theta_f^{(j)} - \rho \frac{\partial E}{\partial \theta_f}(\theta_f^{(j)}, \theta_y^{(i)}, \theta_d^{(j)}), \tag{3.21}$$

$$\theta_y^{(j+1)} = \theta_y^{(j)} - \rho \frac{\partial E}{\partial \theta_y}(\theta_f^{(i)}, \theta_y^{(j)}, \theta_d^{(j)}), \tag{3.22}$$

$$\theta_d^{(j+1)} = \theta_d^{(j)} + \rho \frac{\partial E}{\partial \theta_d}(\theta_f^{(j)}, \theta_y^{(j)}, \theta_d^{(j)}), \tag{3.23}$$

where $\rho > 0$ is the learning rate. Note that equations (3.21) and (3.22) correspond to gradient descent (because we are minimizing E w.r.t. θ_f and θ_y), while equation (3.23) corresponds to gradient ascent (because we are maximizing E w.r.t. θ_d).

Most deep learning libraries only implement minimization algorithms, but the update rule in (3.23) could be implemented by instantiating a new optimizer (only responsible for updating θ_d) with a negative learning rate. An alternative to this is the usage of a gradient reversal layer.

Computing the gradients of E with respect to each parameter and replacing in equations (3.21)-(3.23) yields:

$$\theta_f^{(j+1)} = \theta_f^{(j)} - \rho \left(\frac{1}{n} \sum_{i=1}^{n} \frac{\partial L_y}{\partial \theta_f}(\theta_f^{(j)}, \theta_y^{(j)})(x_i^s) - \frac{\mu}{n} \sum_{i=1}^{n} \frac{\partial L_d}{\partial \theta_f}(\theta_f^{(j)}, \theta_d^{(j)})(x_i^s) - \frac{\mu}{n'} \sum_{i=1}^{n'} \frac{\partial L_d}{\partial \theta_f}(\theta_f^{(j)}, \theta_d^{(j)})(x_i^t) \right), \tag{3.24}$$

$$\theta_y^{(j+1)} = \theta_y^{(j)} - \frac{\rho}{n} \sum_{i=1}^{n} \frac{\partial L_y}{\partial \theta_y}(\theta_f^{(j)}, \theta_y^{(j)})(x_i^s), \tag{3.25}$$

$$\theta_d^{(j+1)} = \theta_d^{(j)} - \rho \left(\frac{\mu}{n} \sum_{i=1}^{n} \frac{\partial L_d}{\partial \theta_d}(\theta_f^{(j)}, \theta_d^{(j)})(x_i^s) + \frac{\mu}{n'} \sum_{i=1}^{n'} \frac{\partial L_d}{\partial \theta_d}(\theta_f^{(j)}, \theta_d^{(j)})(x_i^t) \right). \tag{3.26}$$

From these equations, we observe that:

i optimizing θ_y is achieved by performing gradient descent over the task loss L_y;

ii optimizing θ_d is achieved by performing gradient descent over the domain discrimination loss L_d;

iii optimizing θ_f is achieved by performing gradient descent over the task loss L_y and gradient ascent over the domain discrimination loss L_d.

If we introduce a gradient reversal layer with activation function g at the output of the feature extractor, such that $g(G_f) = G_f$ and $\frac{\tilde{\partial} g}{\partial G_f} = -1$ and use it at the input of the domain discriminator, then:

$$\frac{\tilde{\partial} L_d}{\partial \theta_f} = \frac{\tilde{\partial} L_d}{\partial G_f}\frac{\tilde{\partial} G_f}{\partial \theta_f} \tag{3.27}$$

$$= -\frac{\partial L_d}{\partial G_f}\frac{\partial G_f}{\partial \theta_f} \tag{3.28}$$

$$= -\frac{\partial L_d}{\partial \theta_f} \tag{3.29}$$

Replacing (3.30)-(3.32) with the new manually defined derivatives yields:

$$\theta_f^{(j+1)} = \theta_f^{(j)} - \rho\left(\frac{1}{n}\sum_{i=1}^{n}\frac{\tilde{\partial} L_y}{\partial \theta_f}(\theta_f^{(j)}, \theta_y^{(j)})(x_i^s) + \frac{\mu}{n}\sum_{i=1}^{n}\frac{\tilde{\partial} L_d}{\partial \theta_f}(\theta_f^{(j)}, \theta_d^{(j)})(x_i^s) + \frac{\mu}{n'}\sum_{i=1}^{n'}\frac{\tilde{\partial} L_d}{\partial \theta_f}(\theta_f^{(j)}, \theta_d^{(j)})(x_i^t)\right), \tag{3.30}$$

$$\theta_y^{(j+1)} = \theta_y^{(j)} - \frac{\rho}{n}\sum_{i=1}^{n}\frac{\tilde{\partial} L_y}{\partial \theta_y}(\theta_f^{(j)}, \theta_y^{(j)})(x_i^s), \tag{3.31}$$

$$\theta_d^{(j+1)} = \theta_d^{(j)} - \rho\left(\frac{\mu}{n}\sum_{i=1}^{n}\frac{\tilde{\partial} L_d}{\partial \theta_d}(\theta_f^{(j)}, \theta_d^{(j)})(x_i^s) + \frac{\mu}{n'}\sum_{i=1}^{n'}\frac{\tilde{\partial} L_d}{\partial \theta_d}(\theta_f^{(j)}, \theta_d^{(j)})(x_i^t)\right). \tag{3.32}$$

We can now simply apply gradient descent w.r.t. all parameters of the following loss function:

$$\tilde{E}(\theta_f, \theta_y, \theta_d) = \frac{1}{n}\sum_{i=1}^{n} L_y(\theta_f, \theta_y) + \mu\left(\frac{1}{n}\sum_{i=1}^{n} L_d(\theta_f, \theta_d) + \frac{1}{n'}\sum_{i=1}^{n'} L_d(\theta_f, \theta_d)\right) \tag{3.33}$$

Therefore, by using the gradient reversal layer at the input of the domain discriminator, our initial problem, that involved both gradient descent and ascent over parameters, was cast into a new problem that has a much more canonical form, where a single standard optimizer can be used.

3.5.2 Multisource Adversarial Domain Adaptation

3.5.2.1 Generalization bound for Multiple Source Domain Adaptation

This theorem follows as a generalization from the single-source case, having been proved by Zhao et al. [14]

Theorem 3.5.2. *Consider a hypothesis class H of VC dimension d, k source domains $D_{s_1}, D_{s_2}, ..., D_{s_k}$, from which n samples were drawn: $S_1, S_2, ..., S_k$, and a target domain D_t from which a set T with*

$n \times k$ samples was drawn, then, with probability at least $1-\delta$ over the choice of samples, for every $\eta \in H$ and for every linear combination $\alpha \in \mathbb{R}^k_+$, $\sum_{i=1}^k \alpha_i = 1$:

$$\varepsilon_{D_t}(\eta) \leq \sum_{i=1}^{k} \alpha_i \left(\widehat{\varepsilon_{S_i}}(\eta) + \widehat{d_H}(S_i, T)\right) + \beta_\alpha + O\left(\sqrt{\frac{1}{kn}(\log\frac{1}{\delta} + d\log\frac{kn}{d})}\right) \tag{3.34}$$

with $\beta_\alpha \geq \inf_{\eta* \in H}[\varepsilon_{D_{s_\alpha}}(\eta*) + \varepsilon_{D_t}(\eta*)]$, where $D_{s_\alpha} = \sum_{i=1}^k \alpha_i D_{s_i}$

Like in the single-source case, for minimizing the target risk, both the source risk and the empirical H-divergence should be minimized. Additionally, a linear combination α of the source domains should be chosen, so as to generate a new source domain D_{s_α}, that can minimize β_α.

Developing systematic ways of combining multiple source domains to approximate the target domain is the purpose of multisource domain adaptation algorithms. Hoffman et al. presented an algorithm that learns a convex combination rule to combine the classifiers of each source domain, casting the solution of finding the optimal combination of source domains as a DC-programming problem (difference of convex) [88]. Guo et al. proposed an ensemble of experts approach for unsupervised domain adaptation from multiple sources. For each target sample, the predicted posterior is a weighted combination of all the experts prediction. The weights reflect the proximity of the sample to each source domain [89]. Kim et al. also proposed an ensemble of experts algorithm, that learns example-to-domain relations to weigh each source domain differently for each target example. Additionally, this algorithm does not require a fixed number of domains throughout training, being able to quickly generalize to a new domain with limited supervision [90]. Lastly, Zhao et al. focused on an adversarial approach [14]. We will discuss this approach in more detail in the next section, since our work is based on it.

3.5.2.2 Multisource Domain Adversarial Networks

Zhao et al. proposed an algorithm named Multisource Domain Adversarial Network (MDAN), which is a generalization of the work of Ganin to the multisource case[14], that is, with multiple source domains. The domain-discriminating task is now performed by using one binary classifier per source domain. Figure 3.4 shows the model proposed by Zhao et al.

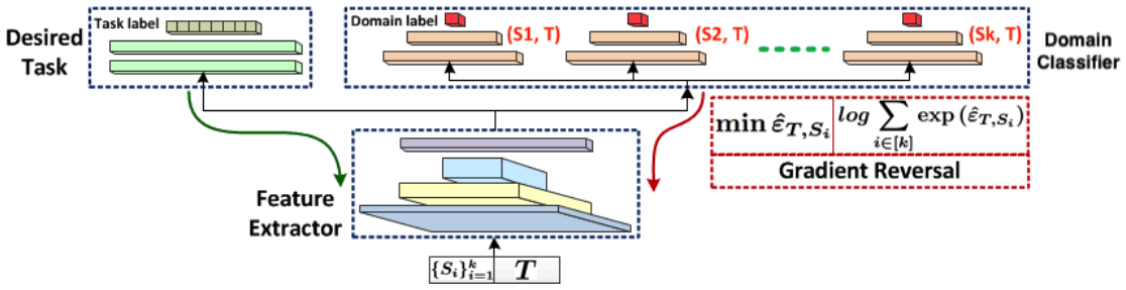

Figure 3.4: MDAN Network architecture. Reprinted from Zhao et al. [14]

The model can be summarized as follows. Assume we have annotated samples $\{(x_i^s, y_i^s)\}_j^{n_j}$ from k source domains $D_{s_1}, D_{s_2}, ..., D_{s_k}$ and unlabeled samples $\{x_i^t\}^{n'}$ from the target domain D_t. Let $G_f(\cdot, \theta_f)$, $G_y(\cdot, \theta_y)$ and $G_d(\cdot, \theta_{d,j})$ denote the feature extractor, the task classifier and the j-th domain discriminator (for $j = 1, 2, ..., k$), respectively. The j-th domain discriminator seeks to classify a sample in the new feature space as part of domain D_{s_j} or D_t. The prediction loss for a given sample x_i is defined as $L_y(\theta_f, \theta_y)(x_i) = L_y(G_y(G_f(x_i, \theta_f), \theta_y), y_i)$ and the domain loss for a given sample x_i is defined as $L_d(\theta_f, \theta_d)(x_i) = L_d(G_d(G_f(x_i, \theta_f), \theta_d), d(x_i))$, where L_y and L_d are the standard classification losses for the learning task and domain discrimination task, respectively, and $d(x_i)$ is 1 if x_i is a source example and 0 if it is a target one.

MDAN seeks the parameters θ_f, θ_y and $\theta_{d,j}$, $j = 1, 2, ..., k$, that solve the following optimization problem:

$$\min_{\theta_f, \theta_y} \max_{j \in \{1,...,k\}} \left[\frac{1}{n_j} \sum_{i=1}^{n_j} L_y(\theta_f, \theta_y)(x_{ij}^s) - \mu \min_{\theta_{d,j}} \left(\frac{1}{n_j} \sum_{i=1}^{n_j} L_d(\theta_f, \theta_d)(x_{ij}^s) + \frac{1}{n'} \sum_{i=1}^{n'} L_d(\theta_f, \theta_d)(x_i^t) \right) \right] \tag{3.35}$$

where $\mu > 0$ is a hyperparameter that controls the weight to be given to the domain discriminator loss when compared to the task classification loss. This is a straightforward extension of the single source problem, where the multisource setting is addressed by taking the hardest source domain at each step of the algorithm, that is, the one that produces the maximum loss.

The authors also presented an alternative formulation where the hard-max over the k source domains is replaced by a soft-max, so that the optimization problem now becomes:

$$\min_{\theta_f, \theta_y} \frac{1}{\gamma} \log \left[\sum_{j=1}^{k} \exp \left(\gamma \left[\frac{1}{n_j} \sum_{i=1}^{n_j} L_y(\theta_f, \theta_y)(x_{ij}^s) - \mu \min_{\theta_{d,j}} \left(\frac{1}{n_j} \sum_{i=1}^{n_j} L_d(\theta_f, \theta_d)(x_{ij}^s) + \frac{1}{n'} \sum_{i=1}^{n'} L_d(\theta_f, \theta_d)(x_i^t) \right) \right] \right) \right] \tag{3.36}$$

where $\gamma > 0$ is a hyperparameter that controls the relative weight to be given to the domains with worse task classification loss, so that the higher this hyperparameter is, the closer it will be to the hard-max version.

The hard-max version has the disadvantage that, if one source domain differs significantly from the target domain when compared to the other source domains, the model will spend an excessive amount of training time focusing only on optimizing that source domain. The latter approach, besides providing a way to combine information from all the source domains, also smooths the objective function. Unsurprisingly, it yields slightly better results in most experiments.

When applied to a multisource setting, our work in this dissertation is based on MDAN. We made this choice as it showed superior performance when compared to other state-of-the-art methods on the task of counting vehicles in images obtained from city cameras videos, a learning task that mirrors the learning process from a sensor network, even though the temporal component of data was not taken into consideration.

Chapter 4

Domain Adaptation for Counting Objects in Video Frames

In this chapter we propose three different domain adaptation methods that take into consideration the temporal and sequential nature of data and one method that does not. The task they execute is the counting of objects, like vehicles or people, in frames extracted from videos from multiple cameras. Thus, the difficulty in this domain adaptation setting lies in adapting frames obtained from one camera to frames obtained from another camera.

We chose to tackle the issue of counting objects from video frames as that is a challenging task related to the application of machine learning techniques to sensor networks. Hence, if the domain adaptation methods we propose are able to accurately count the number of objects in a given frame, we can be confident they will generalize well to other tasks related to sensor networks.

Our methods were all inspired by the network for counting objects FCN-rLSTM, proposed by Zhang et al. [10] and described in more detail in section 2.4 and the adversarial domain adaptation model MDAN, proposed by Zhao et al. [14] and described in more detail in section 3.5.2.2. As the MDAN algorithm does not take into account the temporal and sequential nature of data, we want to investigate if the algorithm can be further improved by leveraging this information in the DA problem.

The implementation of all methods described in this chapter is available at: https://github.com/francis-andrade/domain-adaptation-lstm.

4.1 Basics

In this section we introduce some common notation, definitions and components used to describe our models.

4.1.1 Assumptions

The following assumptions will be made throughout this chapter:

- x is the input variable, representing a particular frame. If x is part of the source domain it is denoted as x_s, whereas if part of a target domain, it is denoted as x_t.
- $density(x)$ is the real density map of objects in frame x.
- $count(x)$ is the real number of objects in frame x. Note that $count(x) = \sum density(x)$.
- $density\_pred(x)$ is the density map predicted by the model for sample x.
- $count\_pred(x)$ is the number of objects predicted by the model for sample x.
- $P(X)$, $P(density(X))$ represent the distributions over frames and densities, respectively.
- $P(X, density(X))$ represents the joint distribution of X and $density(X)$.
- $\{D_{s_1}, D_{s_2}, ..., D_{s_k}\}$ are the joint distributions of X and $density(X)$ in the source domains and $\{D_t\}$ is the joint distribution of x and $density(x)$ in the target domain. If there is only one source domain, then $\{D_s\}$ is the joint distribution of X and $density(X)$ in that source domain.
- $S_1, S_2, ...S_k$ are sets of n samples drawn from domains $D_{s_1}, D_{s_2}, ..., D_{s_k}$, respectively, and T is a set of n samples extracted from the target domain D_t.
- The sets of samples drawn from the domains to be fed into our models have all the same size n.
- In the case of models that take into account the temporal component of frames, a sequence of frames x_seq of size seq_size is fed into the models: $(x_1, x_2, ..., x_{seq\_size})$, such that, given, two frames x_t, x_{t+1}, they must belong to the same video and they must be consecutive frames, that is, x_{t+1} comes after x_t and there can be no frame in the same video that comes after x_t and before x_{t+1}. If x_seq is part of the source domain it is denoted as x_seq_s, whereas if part of the target domain it is denoted as x_seq_t.
- $S\_seq_1, S\_seq_2, ...S\_seq_k$ are sets of n samples of sequences with size seq_size drawn from domains $D_{s_1}, D_{s_2}, ..., D_{s_k}$, respectively, and T_seq is a set of n samples with size seq_size extracted from the target domain D_t.
- Every function f that takes a frame x as input, can also be defined for sequence $x\_seq = x_1, x_2, ..., x_{seq\_size})$ as $f(x\_seq) = (f(x_1), f(x_2), ..., f(x_{seq\_size}))$.
- Let $G_f(\cdot, \theta_f)$, $G_y(\cdot, \theta_y)$ and $G_d(\cdot, \theta_{d,j})$ denote the feature extractor, the task executor and the j-th domain discriminator (for $j = 1, 2, ..., k$), respectively.
- In models where the task executor computes the predicted density map or the object count, $G_{y_{density}}(\cdot, \theta_y)$ and $G_{y_{count}}(\cdot, \theta_y)$ denote the respective value predictions by this sub-model.

- By convention, $h = G_f(x, \theta_f)$. If x is part of the source domain, h is denoted as h_s, whereas, if part of the target domain, it is denoted as h_t.
- The density prediction loss for a given sample x is $L_{density}(density\_pred(x), density(x))$ $= ||density\_pred(x) - density(x))||_F$, where $||.||_F$ is the Frobenius norm.
- The count prediction loss for a given sample x is $L_{count}(count\_pred(x), count(x))$ $= (count\_pred(x) - count(x))^2$.
- L_y is the combined count and density loss, defined as $L_y(x) =$ $L_{density}(density\_pred(x), density(x)) + \lambda L_{count}(count\_pred(x), count(x))$, where λ is a non-negative scalar hyperparameter that controls the weight to be given to the count loss, relative to the density loss.
- $d(x)$ is such that $d(x)$ is 1 if frame x is part of a source domain and 0 is it is part of the target domain.
- $d\_pred(x)$ is the probability of x being part of a source domain, according to our models.
- $L_d(x) = cross\_entropy(d\_pred(x), d(x))$ $= -d(x) \times \log(d\_pred(x)) - (1 - d(x)) \log(1 - d\_pred(x))$ is the domain discrimination loss for sample x.
- μ is a non-negative scalar hyperparameter that controls the weight to be given to the domain discrimination loss L_d relative to the combined count and density loss L_y.
- There is a gradient reversal layer at the input of the domain discriminator.
- $\tilde{E}(\theta_f, \theta_y, \theta_d)$ is the global loss function the model uses for backpropagation. Note that due to the usage of a gradient reversal layer, the minimization of $\tilde{E}$ is not the real optimization problem our model solves. Recall section 3.5.1.3 for a more detailed explanation.

4.1.2 Optimization Problem

We now present the optimization problem for the single source setting and three different optimization problems for the multi-source scenario. Two of the formulations for the multisource scenario, hard-max and soft-max, had already been proposed in the original MDAN paper by Zhang et al. [14].

4.1.2.1 Single source

The optimization problem is:

$$\min_{\theta_f, \theta_y} \left[\frac{1}{n} \sum_{x^s \in S} L_y(x^s) - \mu \min_{\theta_{d,j}} \left(\frac{1}{n} \sum_{x^s \in S} L_d(x^s) + \frac{1}{n} \sum_{x^t \in T} L_d(x^t) \right) \right] \tag{4.1}$$

and, by using a gradient reversal layer at the input of the domain discriminator, the global loss function for backpropagation becomes:

$$\tilde{E}(\theta_f,\theta_y,\theta_d)=\frac{1}{n}\sum_{x^s\in S}L_y(x^s)+\mu\left(\frac{1}{n}\sum_{x^s\in S}L_d(x^s)+\frac{1}{n}\sum_{x^t\in T}L_d(x^t)\right) \tag{4.2}$$

4.1.2.2 Multisource Hard-max

In this case, we intend to solve the following optimization problem:

$$\min_{\theta_f,\theta_y}\max_{j\in\{1,\ldots,k\}}\left[\frac{1}{n}\sum_{x^s\in S_j}L_y(x^s)-\mu\min_{\theta_{d,j}}\left(\frac{1}{n}\sum_{x^s\in S_j}L_d(x^s)+\frac{1}{n}\sum_{x^t\in T}L_d(x^t)\right)\right] \tag{4.3}$$

and will use the following global loss function for backpropagation:

$$\tilde{E}(\theta_f,\theta_y,\theta_d)=\max_{j\in\{1,\ldots,k\}}\left(\frac{1}{n}\sum_{x^s\in S_j}L_y(x^s)\right)+\mu\min_{j\in\{1,\ldots,k\}}\left(\frac{1}{n}\sum_{x^s\in S_j}L_d(x^s)+\frac{1}{n}\sum_{x^t\in T}L_d(x^t)\right) \tag{4.4}$$

4.1.2.3 Multisource Soft-Max

In this case, the optimization problem is:

$$\min_{\theta_f,\theta_y}\frac{1}{\gamma}\log\left[\sum_{j=1}^{k}\exp\left(\gamma\left[\frac{1}{n}\sum_{x^s\in S_j}L_y(x^s)-\mu\min_{\theta_{d,j}}\left(\frac{1}{n}\sum_{x^s\in S_j}L_d(x^s)+\frac{1}{n}\sum_{x^t\in T}L_d(x_i^t)\right)\right]\right)\right] \tag{4.5}$$

whereas the global loss function for backpropagation is:

$$\tilde{E}(\theta_f,\theta_y,\theta_d)=\frac{1}{\gamma}\log\left[\sum_{j=1}^{k}\exp\left(\gamma\left[\frac{1}{n}\sum_{x^s\in S_j}L_y(x^s)+\mu\left(\frac{1}{n}\sum_{x^s\in S_j}L_d(x^s)+\frac{1}{n}\sum_{x^t\in T}L_d(x_i^t)\right)\right]\right)\right] \tag{4.6}$$

4.1.2.4 Multisource Average

This was a new optimization formulation that we introduced for the multisource scenario, in order to give the same weight to every domain. The optimization problem can be formulated as:

$$\min_{\theta_f,\theta_y}\frac{1}{k}\sum_{j=1}^{k}\left[\frac{1}{n}\sum_{x^s\in S_j}L_y(x^s)-\mu\min_{\theta_{d,j}}\left(\frac{1}{n}\sum_{x^s\in S_j}L_d(x^s)+\frac{1}{n}\sum_{x^t\in T}L_d(x^t)\right)\right] \tag{4.7}$$

and the global loss function for backpropagation is:

$$\tilde{E}(\theta_f,\theta_y,\theta_d)=\frac{1}{k}\sum_{j=1}^{k}\left(\frac{1}{n}\sum_{x^s\in S_j}L_y(x^s)\right)+\mu\frac{1}{k}\sum_{j=1}^{k}\left(\frac{1}{n}\sum_{x^s\in S_j}L_d(x^s)+\frac{1}{n}\sum_{x^t\in T}L_d(x^t)\right) \tag{4.8}$$

4.1.3 Component F-HAC

The F-HAC is a component of our models that is the first part of the FCN-rLSTM network, being an intermediate step between the original frame and the predicted density map. Figure 4.1 shows the architecture of F-HAC.

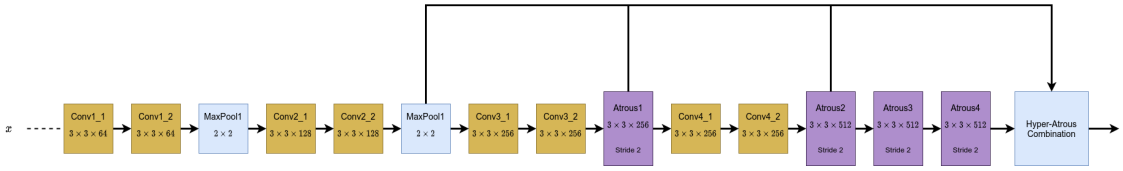

Figure 4.1: F-HAC component

F-HAC starts by applying kernels of size 3×3 to both convolution and deconvolution layers. The number of filter channels in the higher layers is then increased to compensate the information loss caused by the MaxPool layers. The last layer is a hyper-atrous combination that combines the outputs from the second MaxPool layer and the atrous convolution layers.

4.2 Simple Model

The Simple Model was based in a network named FCN-HA proposed by Zhang et al. in the same article that they proposed the FCN-rLSTM [10]. It consists in a simplification of the latter, by not using LSTMs or making any other consideration on the temporal or sequential nature of data. Figure 4.2 shows the architecture of the Simple Model.

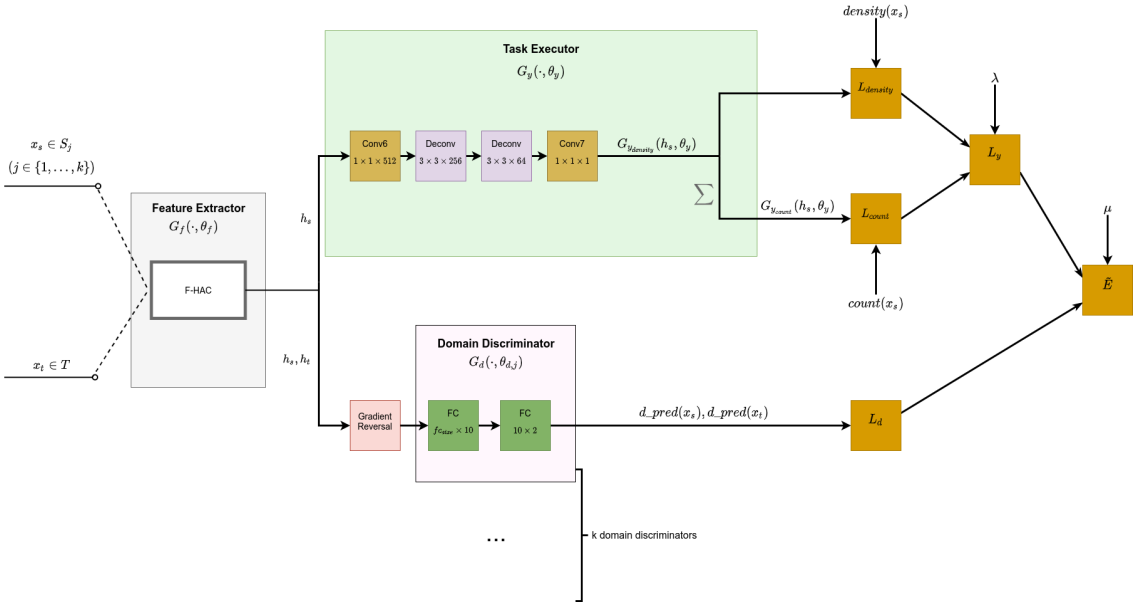

Figure 4.2: Simple Model. fc_{size} is dependent on the dimensions of the frames.

The component F-HAC was used as a feature extractor, whereas the rest of the network FCN-HA was used for the task execution. We opted for choosing the output of the hyper-atrous combination layer to split the network between feature extraction and task execution. As that output is also the input to the last feature reweighting, it guarantees that a common feature representation for the source and target domains will be found while still allowing for some discriminative features between domains to be computed in that last feature reweighting.

For the domain discriminator, we chose two fully connected layers, as we believed it would strike the perfect balance between a domain discriminator too simple where it would be trivial for the feature extractor to fool it and a domain discriminator too complex where it would be too hard for the feature extractor to fool it.

4.3 SingleLSTM Model

The SingleLSTM Model is an extension of the Simple Model, being based in the FCN-rLSTM network proposed by Zhang et al. [10]. Figure 4.3 shows the architecture of the SingleLSTM Model.

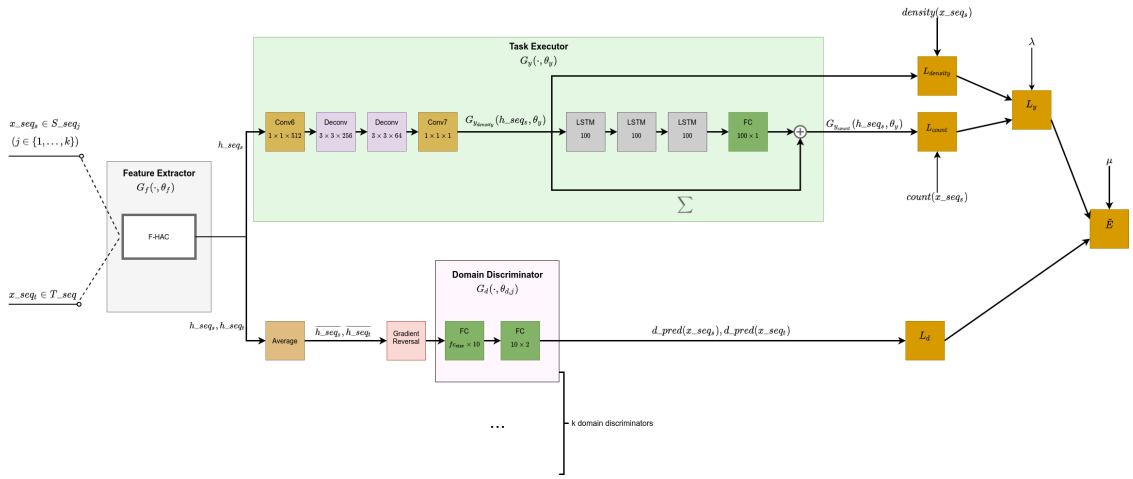

Figure 4.3: SingleLSTM Model. fc_size is dependent on the dimensions of the frames.

Like in the Simple Model case, we chose the output of the hyper-atrous combination to divide the network between feature extractor and task executor, as that layer comes late enough in the network to allow the learning of a feature representation common to all domains and early enough to still allow the model to produce features discriminative enough for executing the learning task.

For domain discrimination, we compute, for each sequence x_seq, an average of the results of the hyper-atrous combination and then, like in the Simple Model case, use two fully connected layers. As the task executor considers the temporal nature of data, but the domain discriminator does not, there is the risk that the task execution loss L_y will be considerably lower than the domain discrimination loss L_d at the beginning of the training process. This mismatch would result in the model giving too much importance to the domain discrimination loss relative to the task execution

loss, which could affect the stability of the model in a later phase of training, as it would likely cause peaks in the task execution loss.

4.4 DoubleLSTM Model

The DoubleLSTM Model is similar to the SingleLSTM Model, being also inspired by the FCN-rLSTM network, proposed by Zhang et al. [10]. Figure 4.4 shows the architecture of the DoubleLSTM Model.

The difference between the DoubleLSTM and the SingleLSTM lies in the domain discriminator, that in this case also considers the temporal nature of data by using an LSTM. In this model, the domain discriminator consists in two deconvolution layers that reduce the spatial size of h_seq_s and h_seq_t, followed by three LSTM layers and two fully connected layers.

By using three LSTM layers for domain discrimination we solve one potential issue of the SingleLSTM model, the mismatch of the task executor and domain discriminator loss at the beginning of training. At the same time, we also risk having an exceedingly complex domain discriminator, which could make it hard for the feature extractor to fool it.

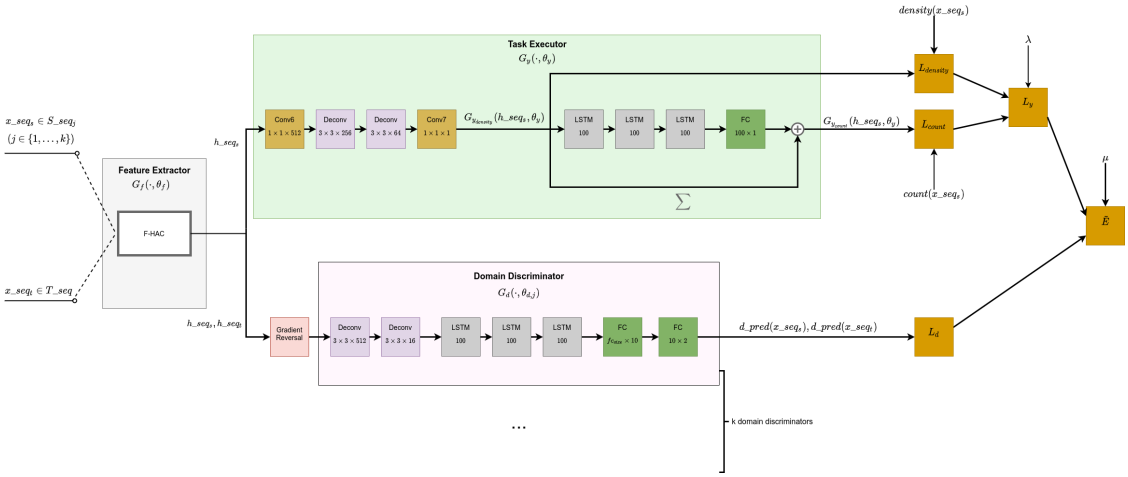

Figure 4.4: DoubleLSTM Model. $fc_{size} = 100 \times seq\_size$.

4.5 CommonLSTM Model

Despite also being inspired by the FCN-rLSTM [10], this model differs significantly from the other two temporal models. Figure 4.5 shows the architecture of the common model.

Unlike the previous models, we do not choose the output of the F-HAC for the split between the feature extractor and the domain discriminator. Instead, we use the output of the last LSTM layer to make this split. The task executor $G_{y_{count}}(h\_seq_s, density\_pred(x\_seq_s), \theta_y)$ takes an extra argument $density\_pred(x\_seq_s)$, unlike the other models that are only dependent on h_seq_s and

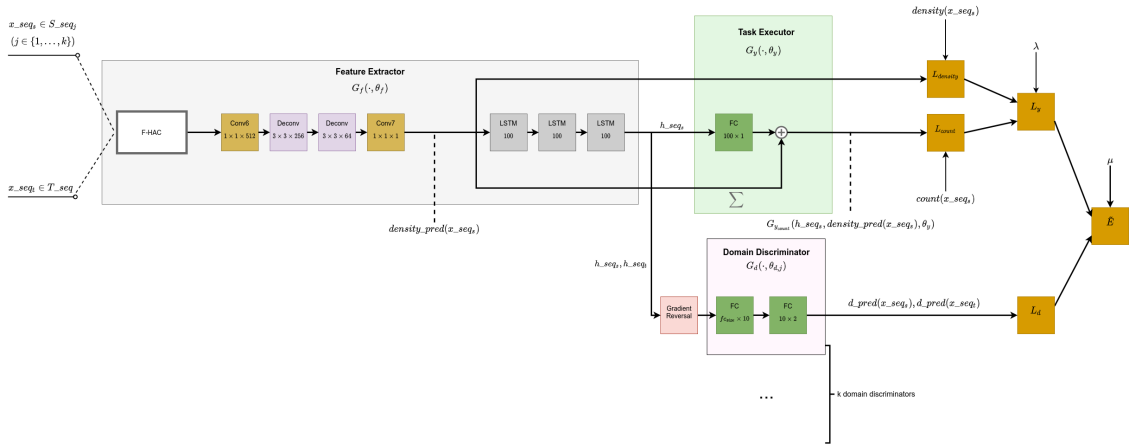

Figure 4.5: CommonLSTM Model. $fc_{size} = 100 \times seq\_size$.

θ_y. Also unlike the other models, it is not the job of the task executor to predict the density map but that of the feature extractor.

For the domain discriminator we use the same architecture used in the Simple and SingleLSTM models with two fully connected networks.

The fact that we choose the split between feature extractor and task executor to be so late in the network has the advantage of allowing the feature extractor to make temporal considerations when mapping the frames into a new feature space. At the same time, it also risks forcing the predicted density maps of source and target domains to be exceedingly similar, not allowing the accurate counting of the number of objects.

4.6 Summary

All the models proposed have specific strengths and weaknesses that would make them suitable for certain scenarios and unlikely to show a good performance in others. As the networks of a DA model are strongly interconnected, small changes in the architecture of a model can cause big shifts in its results.

If the frame rate per second of a video is large and the number of objects in a given frame is not a good predictor for the number of objects in a subsequent frame, the Simple Model is ideal, as it does not make any temporal consideration. In this case, using a temporal model would probably just upset the training process.

If there is a weak but noticeable correlation of the object count in consecutive video frames, the SingleLSTM is likely the best choice. By using an LSTM for the task executor but none for the domain discriminator, it will be able to make small adjustments in the predicted object count, but not let the temporal correlation between frames determine the domain-invariant feature representation generated.

When there is a strong correlation of the object count in consecutive video frames, models DoubleLSTM or CommonLSTM are likely to show the best performance. Their approach differs,

as model DoubleLSTM uses an LSTM layer for the task executor and another one for the domain discriminator, whereas CommonLSTM includes an LSTM layer in the feature extractor. As the CommonLSTM model will indirectly enforce similarity between the predicted density maps, it is probably the best choice when the difference between domains is small. DoubleLSTM, on the other hand, would likely be the best solution when there are significant dissimilarities between domains.

Chapter 5

Experiments

In this chapter, we present the experiments that we ran in order to validate our models. We ran experiments on a crowd counting dataset UCSPeds [15] and on a vehicle counting dataset WebCamT [16].

5.1 Scenario

We will run experiments with networks FCN-HA and FCN-rLSTM, proposed by Zhang et al. in paper "FCN-rLSTM: Deep Spatio-Temporal Neural Networks for Vehicle Counting in City Cameras" [10], described in detail in section 2.4. The FCN-HA does not consider the temporal nature of data, whereas the FCN-rLSTM does. Additionally, we will run experiments with the new domain adaptation methods proposed in this dissertation: Simple Model, SingleLSTM Model, DoubleLSTM Model and CommonLSTM Model, described in detail in chapter 4. Recall that Simple Model does not make temporal considerations, unlike the other three.

In every experiment, assume we have k domains $D_1, D_2, ..., D_k$. In a domain adaptation scenario, we will do k runs, so that in run number t, domain D_t will be chosen as the target domain and all the others will be source domains. In a non-domain adaptation scenario, we also do k runs, so that in run t, D_t will be used for validation/testing and all the other domains will be used for training.

For every experiment, we have computed two types of results, unsupervised and semi-supervised, that we describe below:

- **Unsupervised:** In this case, we do not have a validation dataset, and, for each run t, we calculate the testing results with the model we obtained at the end of the training epochs, for all the samples extracted from domain D_t.

- **Semi-supervised:** For semi-supervised results, we have a validation dataset, that corresponds to 30% of the samples extracted from target D_t. We use those samples to select the

best epoch obtained throughout training. The model obtained in that epoch will be used to test the remaining 70% of the samples in D_t.

For each dataset, we ran a total of 5 experiments for the DA models, where we varied the value of hyperparameter μ between the values [1e-5, 1e-4, 1e-3, 1e-2, 1e-1]. Recall that μ controls the weight to give to the domain discrimination loss, relative to the task execution loss. For the non-DA models, we ran just 1 experiment.

In all experiments, we train the model for 50 epochs, using an Adam optimizer with a learning rate of 1e-4. The adopted evaluation metric is the Mean Absolute Error (MAE) between the predicted object count and the ground truth for each frame.

5.2 UCSPeds

UCSPeds [15] is a crowd counting dataset that contains videos of pedestrians on University of California, San Diego walkways, taken from a stationary camera in two different viewpoints. All videos are 8-bit grayscale, with dimensions 158×238 at a 2 frames per second rate. For each video frame, the dataset has a mask indicating the region of interest in the image and another file indicating the centers, in image coordinates, of every person in the frame.

Each camera point of view corresponds to a domain in our experiments that are named: *vidd* and *vidf*. Figures 5.1-5.2 show examples of frames in each one of the two domains.

Figure 5.1: Domain *vidd* from UCSPeds dataset [15].

Figure 5.2: Domain *vidf* from UCSPeds dataset [15].

Table 5.1 shows the mean and standard deviation for the number of people in a frame, for each domain. There is a strong target shift between domains as the mean number of people in domain *vidf* is more than 4 times higher than the mean number of people in domain *vidd* and, therefore, the DA task is challenging for this dataset.

Domain	**Mean**	**Std**
vidd	6.448	3.089
vidf	27.570	7.393

Table 5.1: UCSPeds domains mean and standard deviation for the number of people in a frame

For each frame, we computed its density map, by placing a 15×8 Gaussian kernel with sum 1 on the center of each person. Figure 5.3 shows an example of a density map, where the Gaussian kernel around each person is shown in red. The region of interest is also indicated in the figure. As we had only 800 frames in each domain, we performed data augmentation by adding the horizontal flip of each frame to the dataset.

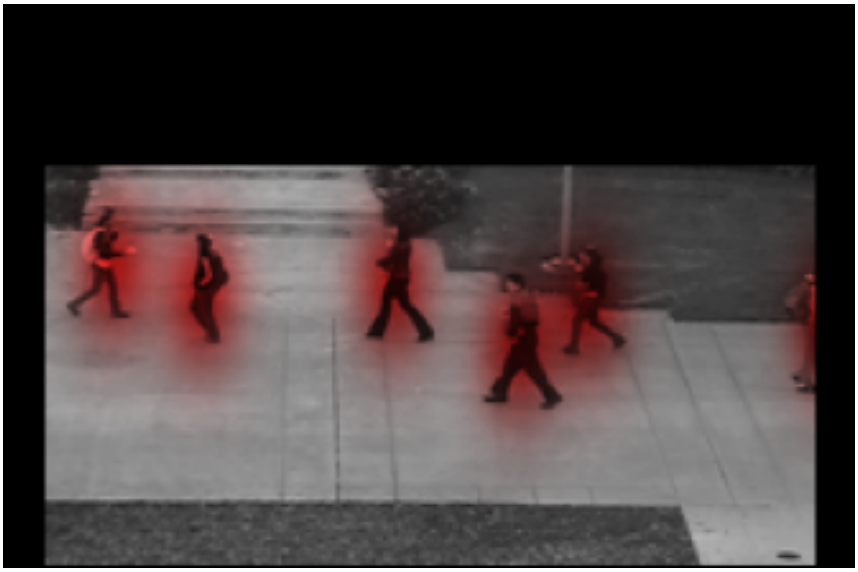

Figure 5.3: Example density map for UCSPeds dataset

5.2.1 Unsupervised Setting

Table 5.2 shows the results in the unsupervised setting. For the domain adaptation models (Simple, SingleLSTM, DoubleLSTM and CommonLSTM), columns *vidd* and *vidf* indicate the best MAE Count obtained in the 5 experiments with the different values of μ indicated before. Column **Avg** indicates the best average of the MAE Count across domains *vidd* and *vidf* in the 5 experiments. That is, for each experiment, we compute the average MAE Count across the 2 domains. We then select the best of the computed average MAE Count.

	MAE Count		
Model	***vidd***	***vidf***	**Avg**
FCN-HA	15.231	7.142	11.186
FCN-rLSTM	11.039	5.981	8.510
Simple	**6.237**	22.517	14.377
SingleLSTM	7.748	9.072	9.316
DoubleLSTM	8.041	**3.893**	6.489
CommonLSTM	6.828	4.795	**6.178**

Table 5.2: MAE Count by domain. For each domain, the best MAE obtained from experiments run with different $\mu's$ is shown. Column **Avg** indicates the best average of the MAE Count for domains *vidd* and *vidf* in experiments run with different values of μ.

For the average of the MAE Count across domains, the model CommonLSTM showed the best results with a MAE of 6.178, followed by the model DoubleLSTM with a MAE of 6.489. Any temporal model showed a better MAE Count Avg. than any non-temporal model, which validates our initial hypothesis that making temporal considerations would significantly improve the accuracy of object counting methods.

When it comes to the comparison between DA models and non-DA models, we can verify that DA models performed better, as models DoubleLSTM and CommonLSTM showed the best results for MAE Count Avg. Still, in this case, the advantage of using domain adaptation is not as clear as the advantage of making temporal considerations. For example, the Simple Model, that executes domain adaptation, showed worse results than FCN-HA and FCN-rLSTM. It appears that for domain adaptation to show an improvement in results, it must be used in combination with a temporal model, as otherwise, it will just fluster the training process.

As for the performance of models across different domains, we can verify that they are not consistent. For example, Model Simple showed the best results in domain *vidd*, despite showing the worst results overall. Models DoubleLSTM and CommonLSTM performed particularly well in domain *vidf*.

Graph 5.4 shows the Avg. MAE Count across domains for every DA model experiment, run with a different value of μ.

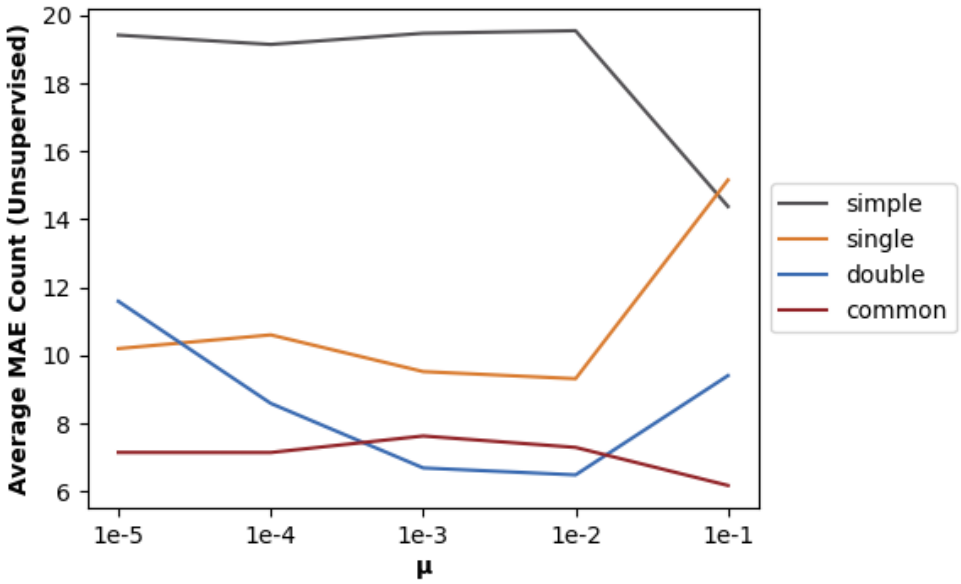

Figure 5.4: Avg. MAE Count across domains for every μ. Unsupervised results.

This graph confirms what we have already verified above, that temporal models perform significantly better than non-temporal ones. Model CommonLSTM, in particular, showed results more than 50% better than Model Simple for every value of μ. When it comes to the best value of μ, we can verify that models DoubleLSTM and SingleLSTM showed the best results for values 1e-2 and 1e-3, whereas models Simple and CommonLSTM showed the best results for value 1e-1. Model SingleLSTM in particular showed significantly worse results for a μ value of 1e-1. As

explained in section 4.3, in model SingleLSTM the task executor is far more complex than the domain discriminator, which can lead to a mismatch of the task execution and domain discrimination losses. The higher the value of μ, the higher is the risk of this mismatch, which explains the worse performance of the SingleLSTM model in this case.

5.2.2 Semi-supervised Setting

Table 5.3 shows the results in the semi-supervised setting. Like in table 5.2, for the domain adaptation models (Simple, SingleLSTM, DoubleLSTM and CommonLSTM), columns *vidd* and *vidf* indicate the best MAE Count obtained in the 5 experiments with the different values of μ indicated before. The column **Avg** indicates the best average of the MAE Count for domains *vidd* and *vidf* in the 5 experiments.

	MAE Count		
Model	***vidd***	***vidf***	**Avg**
FCN-HA	1.432	4.477	2.955
FCN-rLSTM	2.008	**2.074**	2.041
Simple	0.838	3.491	2.209
SingleLSTM	0.942	5.067	3.004
DoubleLSTM	0.952	2.835	**1.893**
CommonLSTM	**0.881**	2.612	1.935

Table 5.3: MAE Count by domain. For each domain, the best MAE obtained from experiments run with different $\mu's$ is shown. Column **Avg** indicates the best average of the MAE Count for domains *vidd* and *vidf* in experiments run with different values of μ.

It is possible to observe that the semi-supervised results are significantly better than the unsupervised ones, with the semi-supervised MAE Count being less than one third of the unsupervised MAE Count, for each model in every domain. As the mean count of the number of people per frame in the two domains differs significantly, it leads to instability in training, which causes the semi-supervised results, where we select the best epoch, to be significantly better than the unsupervised results, where we just select the last one.

We can discern the same general remarks we made when we analysed the unsupervised results. Models DoubleLSTM and CommonLSTM showed the best performance. Temporal models proved better than non-temporal ones, although in this case, model SingleLSTM had the worst MAE Count Avg., and DA models proved, in general, to be better than non-DA models.

Graph 5.5 shows the Avg. MAE Count across domains for every DA model in this experiment, run with a different value of μ. Like in the graph with the unsupervised results, it is possible to verify that the temporal models showed the best results, although for a μ value of 1e-1, the only temporal model that had better results than the Simple Model was the CommonLSTM model. We also notice that the SingleLSTM had a significantly worse result for μ =1e-1.

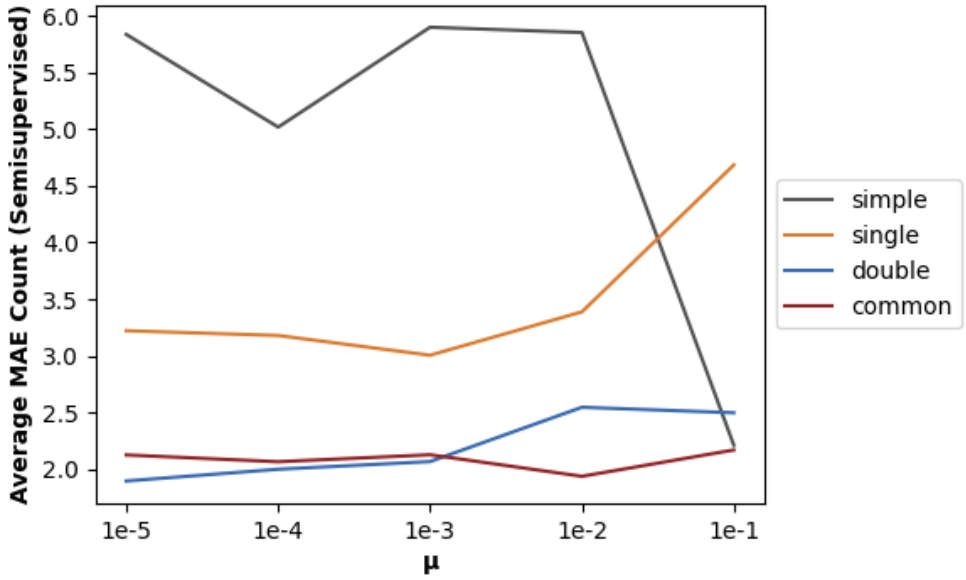

Figure 5.5: Avg. MAE Count across domains for every μ. Semi-supervised experiment.

5.3 WebCamT

WebCamT [16] is a vehicle counting dataset that contains city camera videos, taken from a stationary camera in different points of the city of New York. All videos are in color, with dimensions 240×352, at a 1 frame per second rate. Every frame in the dataset has a ground truth file with annotations that indicate the center of each vehicle and also its bounding box.

There are a total of 14 cameras, that cover multiple scenes, camera perspectives, congestion states and weather conditions. We chose a total of 4 cameras from those 14 to correspond to our domains. Each of those 4 cameras had a number of videos, from which we selected 1000 frames. The frames were selected consecutively, so that if frames f_1 and f_2 from the same video V were selected, then every frame between f_1 and f_2 in video V was also selected.

Figures 5.6-5.9 show examples of frames in each one of the four domains. Table 5.4 shows the mean and standard deviation for the number of vehicles in a frame, for each domain. As it is possible to observe, the mean of vehicles in each frame differs significantly across domains, with the mean number of vehicles in domain *691* being more than 3 times higher than the mean number of vehicles in domain *846*.

Domain	**Mean**	**Std**
511	12.039	6.32
551	18.362	4.21
691	25.470	10.738
846	6.873	2.477

Table 5.4: WebCamT domains mean and standard deviation

After extracting each frame from the videos, we computed its density map, by placing a 4×4

Figure 5.6: Domain *511* from WebCamT dataset [16].

Figure 5.7: Domain *551* from WebCamT dataset [16].

Figure 5.8: Domain *691* from WebCamT dataset [16].

Figure 5.9: Domain *846* from WebCamT dataset [16].

Gaussian kernel with sum 1 on the center of each car. We then had to resize the frames and density to maps to size 120×176 so that we could deal with speed and memory constraints. Figure 5.10 shows an example of a density map in a resized frame, where the Gaussian kernel around each car is shown in red.

As in each run of the experiment there are 3 different source domains, this is a multisource scenario. As such, there are three possible optimization problem formulations for our DA models to solve, described in detail in section 4.1.2 . We decided to go with the **Average** as, given the wide range of values for the mean number of vehicles in each frame across domains, it would be likely for one source domain to show a significantly higher loss than the others. If we used any of the other two optimization problems, our models would dedicate an outweighed importance to the hardest source domain, disregarding the other domains. Table 5.5 shows experiments run comparing how well different optimization problems work with different models, that confirmed the superior performance of the formulation Average in dataset WebCamT.

5.3.1 Unsupervised Setting

Table 5.6 shows the results in the unsupervised setting. For the domain adaptation models (Simple, SingleLSTM, DoubleLSTM and CommonLSTM), columns *511*, *551*, *691* and *846* indicate the best MAE Count obtained for the respective domain in the 5 experiments with the different values

Figure 5.10: Example density map for WebCamT dataset

	Optimization Problem		
Model	**Hardmax**	**Softmax**	**Average**
Simple	18.146	11.160	9.913
SingleLSTM	10.543	9.326	11.101
DoubleLSTM	9.589	7.731	6.148
CommonLSTM	9.847	12.069	10.219

Table 5.5: Comparison of different optimization problems. The table indicates the Avg. MAE Count across domains. The experiments were run with μ =1e-3.

of μ indicated before. Column **Avg** indicates the best average of the MAE Count across domains *511*, *551*, *691* and *846* in the 5 experiments. That is, for each experiment, we compute the average MAE Count across the 4 domains. We then select the best of the computed average MAE Count.

	MAE Count				
Model	***511***	***551***	***691***	***846***	**Avg**
FCN-HA	**5.006**	**2.577**	17.224	4.674	7.370
FCN-rLSTM	6.26	8.789	13.632	6.305	8.746
Simple	9.629	3.497	18.271	5.581	9.846
SingleLSTM	6.579	5.100	15.264	9.553	10.532
DoubleLSTM	5.013	4.576	10.973	**3.029**	**6.148**
CommonLSTM	6.106	5.552	**10.508**	3.564	8.146

Table 5.6: MAE Count by domain. For each domain, the best MAE obtained from experiments run with different $\mu's$ is shown. Column **Avg** indicates the best average of the MAE Count for domains *511*, *551*, *691* and *846* in experiments run with different values of μ.

For the average of the MAE Count across domains, the model DoubleLSTM showed the best results with a MAE of 6.148, followed by the FCN-HA with a MAE of 7.370. In this dataset, the

temporal models did not seem to perform particularly better than the non-temporal ones. Even though the DoubleLSTM showed the best results, the second best was the FCN-HA, a model that does not make any temporal consideration. We also could not verify a marked difference in the MAE Count Avg. of the DA and non-DA methods. These observations ascertain that it is not enough to apply domain adaptation or consider the temporal nature of data to obtain good results. Thus, the careful design of models and its integration with the two mentioned techniques is essential.

When it comes to the accuracy of models across different domains, we find that domain *691* has the worst results, since it has the most dissimilar vehicle count distribution (highest mean and also highest standard deviation), which cannot be matched by any mixture of the source vehicle count distributions. Just like in the UCSPeds dataset, it is observable that the performance of models across different domains is not consistent. For example, the FCN-HA model shows the best results for domains *511* and *551*, but the second worst result for domain *691*.

We find that the model with the most balanced performance across domains was the DoubleLSTM as its MAE Count in a given model was either the best or the second best. This coupled with the fact that the DoubleLSTM was also the model with the best MAE Count Avg. allows us to conclude that this method was the best-performing in the WebCamT dataset.

Graph 5.11 shows the Avg. MAE Count across domains for every DA model experiment, run with a different value of μ.

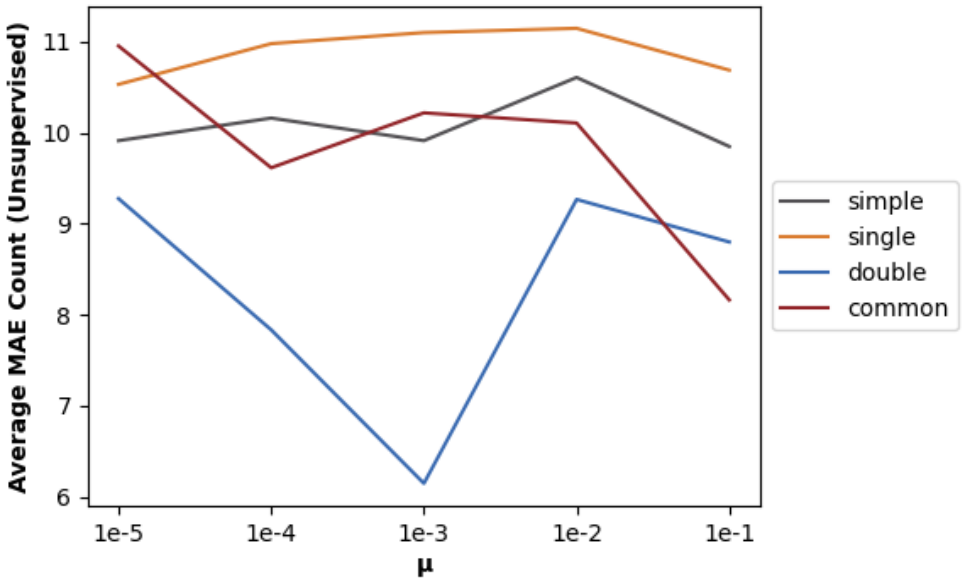

Figure 5.11: Avg. MAE Count across domains for every μ. Unsupervised results.

This graph confirms that the model DoubleLSTM was the best-performing in the WebCamT dataset. The SingleLSTM, on the other hand, showed the worst results, even when comparing it against Simple Model, a non-temporal method. It is also noticeable that model DoubleLSTM has the best accuracy for the intermediate value of μ =1e-3. On the other hand, CommonLSTM shows a tendency to improve the MAE Count as the μ decreases.

5.3.2 Semi-supervised Setting

Table 5.7 shows the results in the semi-supervised setting. Like in the table 5.6, for the domain adaptation models (Simple, SingleLSTM, DoubleLSTM and CommonLSTM), columns *511*, *551*, *691* and *846* indicate the best MAE Count obtained in the 5 experiments with the different values of μ indicated before. The column **Avg** indicates the best average of the MAE Count for domains *511*, *551*, *691* and *846* in the 5 experiments.

	MAE Count				
Model	***511***	***551***	***691***	***846***	**Avg**
FCN-HA	**2.050**	**2.589**	6.560	2.394	**3.398**
FCN-rLSTM	5.400	3.486	9.153	3.537	5.394
Simple	3.112	3.565	10.964	2.049	5.035
SingleLSTM	4.239	4.393	7.032	**1.520**	4.327
DoubleLSTM	4.223	4.552	**6.456**	1.583	4.271
CommonLSTM	4.227	4.543	8.668	1.634	4.984

Table 5.7: MAE Count by domain. For each domain, the best MAE obtained from experiments run with different $\mu's$ is shown. Column **Avg** indicates the best average of the MAE Count for domains *511*, *551*, *691* and *846* in the experiments run with different values of μ.

There is a noticeable difference between the semi-supervised results and the unsupervised ones, with the former being significantly better that the latter. Still, this difference is not as wide as the one verified in the UCSpeds dataset. As the experiments we ran with the WebCamT had more data and more domains, that led to a reduction in the instability of the models, which caused the difference in accuracy between the two settings to not be as wide.

Like in the unsupervised results, we were not able to notice a significant difference between the performance of the temporal methods and the non-temporal ones. As we have also verified before, the DA models do not show a marked improvement over the non-DA models for the WebCamT dataset. In fact, in this case, it was a non-temporal, non-DA model, the FCN-HA, that showed the best MAE Count Avg.

Graph 5.12 shows the Avg. MAE Count across domains for every DA model experiment, run with a different value of μ.

The DoubleLSTM had, in general, the best Avg. MAE Count in the semi-supervised graph, similar to what it was verified in the unsupervised case. The Simple Model was the worst, unlike in the unsupervised setting, where it was the SingleLSTM that showed the worst results.

The Avg. MAE Count across different values of μ is considerably more stable in the semi-supervised results than in the unsupervised ones. We were not able to notice an improvement in the performance of the CommonLSTM as μ decreased, just like we were not able observe a notorious lower Avg. MAE Count in the DoubleLSTM model, for μ =1e-3. Because the unsupervised results are more unstable, they are also more sensible to variations of the value of μ that are not discernible in the semi-supervised graph.

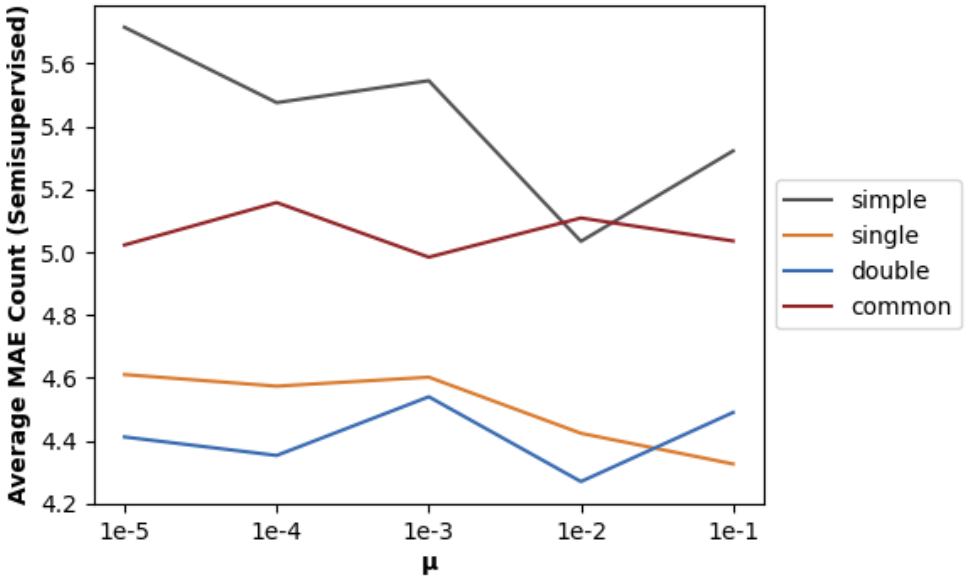

Figure 5.12: Avg. MAE Count across domains for every μ. Semi-supervised experiment.

5.4 Discussion

It stands out that models DoubleLSTM and CommonLSTM showed a better performance than the state-of-art method for counting objects FCN-rLSTM in every dataset and in every setting (unsupervised and semi-supervised). In particular, when it comes to the Avg. MAE Count in the unsupervised results, the DoubleLSTM model showed an average improvement of 27% , whereas the CommonLSTM model showed an average improvement of 17%, when compared against the FCN-rLSTM. As such, these results underline the contribution this dissertation made to the problem of adversarial domain adaptation in sensor networks.

Models Simple and SingleLSTM, on the other hand, had a very unsatisfactory performance, by not showing better results than methods FCN-rLSTM and FCN-HA, even though they leveraged the power of domain adaptation and, in the case of SingleLSTM, the power of LSTMs to make temporal considerations about the nature of data. When it comes to the Simple Model, it appears the reason for its low accuracy is the fact that, without the power of LSTMs, applying domain adaptation to the FCN-HA just disturbs the training process. As for the SingleLSTM, having a too big of a difference in the complexity of its task executor and its domain discriminator is the most likely cause for its failure to show improvements relative to the state-of-art methods.

We have thus verified that integrating domain adaptation with a method does not automatically guarantee better results. Careful considerations on how to divide the previous method between feature extractor and task executor should be made, just like careful deliberations on the domain discriminator design.

It should also be noticed that the results showed an inconsistency in the performance of models across domains. That is, whereas a model performed better in one domain, another model performed better in another. Hence, when applying a model for counting objects in a real world

scenario, one should ponder what model to use depending on the characteristics of the particular target domain.

Chapter 6

Conclusion

This chapter includes our final remarks about the work we developed, where we give an overview of our dissertation, sum up the main contributions of our dissertation and explore future lines of research that our work opened up.

6.1 Overview

Sensor networks, defined as a set of entities somehow related to each other and that produce a stream of data over time, are becoming increasingly common. From city cameras to IOT, these networks have become part of our daily lives.

Still, there is a lack of adequate machine learning methods for predicting their behaviour. That shortage has been even more exacerbated by the surge of big data, with its demand for large datasets that are impossible to manually annotate. Domain adaptation is an answer to the constraints introduced by large partially annotated datasets, since it allows for the transfer of knowledge from labeled samples to unlabelled ones.

Nevertheless, the application of domain adaptation to sensor networks has been seldom studied and, when it was, the temporal and sequential nature of data was not taken into consideration. This dissertation seeks to research how to adapt one of the most advanced DA techniques, adversarial domain adaptation, to effectively consider the sequential and temporal nature of data. We study the specific case of counting objects of a specific category in video frames, since that is a challenging task related to sensor networks and one where large benchmark datasets exist. As such, we can be confident that if our models perform well in this case, they will be able to generalize well to other situations.

For the design of our models, we integrated one state-of-the-art model related to adversarial domain adaptation MDAN with LSTMs, the network of choice for handling temporal data. We further analysed how to combine these two ideas with state-of-art methods for counting objects in

video cameras. In total, we proposed four new methods that leverage the use of domain adaptation and that account for different ways of combining the concepts mentioned above.

Lastly, we presented a thorough evaluation of our models with extensive experiments that reveal what models proposed by us show improvements relative to the most recent alternative methods. We also discuss what prevented some of our models from achieving the desired results.

6.2 Contributions

This dissertation produced the following contributions:

- A survey of domain adaptation methods and the history of the field.
- A detailed review of the technique of adversarial domain adaptation, where, besides the practical explanation of its models, an analysis of the theoretical fundamentals of this area was also included.
- The proposal of a new optimization problem for domain adaptation in a multisource setting, which we named "Multisource Average".
- The proposal of four different models that leveraged domain adaptation for counting objects in video frames. Those models are named "Simple Model", "SingleLSTM", "DoubleLSTM" and "CommonLSTM".
- The evaluation of the performance of our models, that showed model "DoubleLSTM" to have an accuracy 27% better than other the state-of-the-art methods and model "CommonLSTM" to have an accuracy 17% better than other state-of-the-art methods.

6.3 Future Work

This section explains some open lines of research in the field of adversarial domain adaptation, related to the work we developed in this dissertation:

- **Improve the Simple Model**, so that we have a domain adaptation model that is able to count objects in images effectively, without the need for making temporal considerations about data.
- **Improve the SingleLSTM Model**, so that there can be an effective domain adaptation model for counting objects in video frames that does not use an LSTM in the domain discrimination task and, as such, does not make temporal considerations for generating the domain-invariant feature representation. The key to this improvement might be increasing the complexity of the domain discriminator by adding extra layers to it, so as to close the gap in complexity between the domain discriminator and the task executor.

- **Generalize our models for other object counting methods**. All the models proposed by us have the FCN-rLSTM as their basis. It would be interesting to check how well they work with other networks for counting objects. For example, model Deep Recurrent Spatial-Aware Network, proposed by Liu et al. [3] can be the first case of study.
- **Generalize our models for other tasks**, so that it can be proven that they are able to work effectively for any situation related to sensor networks.
- **Improve the proposed methods for combining multiple source domains in the multi-source setting**. Regarding this improvement, one modification that should be studied is the use of a single k-class classifier for the domain discrimination task instead of multiple binary classifiers.

References

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

[2] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, pp. 2222–2232, Oct 2017.

[3] L. Liu, H. Wang, G. Li, W. Ouyang, and L. Lin, "Crowd counting using deep recurrent spatial-aware network," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, p. 849–855, AAAI Press, 2018.

[4] D. Varshneya and G. Srinivasaraghavan, "Human trajectory prediction using spatially aware deep attention models," *CoRR*, vol. abs/1705.09436, 2017.

[5] A. Vo, "Deep learning – computer vision and convolutional neural networks." https://anhvnn.wordpress.com/2018/02/01/deep-learning-computer-vision-and-convolutional-neural-networks/. Accessed: 2020-06-19.

[6] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Computation*, vol. 29, no. 9, pp. 2352–2449, 2017. PMID: 28599112.

[7] C. Jiao, K. Su, W. Xie, and Z. Ye, "Burn image segmentation based on mask regions with convolutional neural network deep learning framework: more accurate and more convenient," *Burns Trauma*, vol. 7, 12 2019.

[8] J. Rocca, "Understanding generative adversarial networks (gans)." https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29. Accessed: 2020-06-21.

[9] R. Tolosana, R. Vera-Rodriguez, J. Fierrez, A. Morales, and J. Ortega-Garcia, "Deepfakes and beyond: A survey of face manipulation and fake detection," *arXiv preprint arXiv:2001.00179*, 2020.

[10] S. Zhang, G. Wu, J. P. Costeira, and J. M. F. Moura, "Fcn-rlstm: Deep spatio-temporal neural networks for vehicle counting in city cameras," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 3687–3696, 2017.

[11] A. Rozantsev, M. Salzmann, and P. Fua, "Beyond sharing weights for deep domain adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, 03 2016.

[12] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *Computer Vision – ECCV 2016 Workshops* (G. Hua and H. Jégou, eds.), (Cham), pp. 443–450, Springer International Publishing, 2016.

[13] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, p. 1180–1189, JMLR.org, 2015.

[14] H. Zhao, S. Zhang, G. Wu, J. M. F. Moura, J. P. Costeira, and G. J. Gordon, "Adversarial multiple source domain adaptation," in *Advances in Neural Information Processing Systems 31* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 8559–8570, Curran Associates, Inc., 2018.

[15] A. B. Chan and N. Vasconcelos, "Counting people with low-level features and bayesian regression," *IEEE Trans. on Image Processing*, vol. 21, no. 4, pp. 2160–77, 2012.

[16] S. Zhang, G. Wu, J. Costeira, and J. Moura, "Understanding traffic density from large-scale web camera data," in *ArXiv*, pp. 4264–4273, 07 2017.

[17] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of Statistical Planning and Inference*, vol. 90, pp. 227–244, 10 2000.

[18] J. Huang, A. Smola, A. Gretton, K. Borgwardt, and B. Schölkopf, "Correcting sample selection bias by unlabeled data.," in *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference, 601-608 (2007)*, vol. 19, pp. 601–608, 01 2006.

[19] I. B. Arief-Ang, F. D. Salim, and M. Hamilton, "Da-hoc: Semi-supervised domain adaptation for room occupancy prediction using co2 sensor data," in *Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments*, BuildSys '17, (New York, NY, USA), Association for Computing Machinery, 2017.

[20] I. B. Arief-Ang, M. Hamilton, and F. D. Salim, "A scalable room occupancy prediction with transferable time series decomposition of co2 sensor data," *ACM Trans. Sen. Netw.*, vol. 14, Nov. 2018.

[21] P. Pinheiro, "Unsupervised domain adaptation with similarity learning," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8004–8013, 06 2018.

[22] P. M. Ferreira, D. Pernes, A. Rebelo, and J. S. Cardoso, "Learning signer invariant representations with adversarial training," in *12th International Conference on Machine Vision (ICMV 2019)*, 2019.

[23] M.-H. Chen, Z. Kira, and G. AlRegib, "Temporal attentive alignment for video domain adaptation," in *International Conference on Computer Vision (ICCV)*, 2019.

[24] Y. Liu and X. Li, "Domain adaptation for land use classification: A spatio-temporal knowledge reusing method," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 98, pp. 133 – 144, 2014.

[25] J. W. Gichoya, S. Nuthakki, P. G. Maity, and S. Purkayastha, "Phronesis of ai in radiology: Superhuman meets natural stupidity," *ArXiv*, vol. abs/1803.11244, 2018.

[26] J. French, "The time traveller's capm," *Investment Analysts Journal*, vol. 46, no. 2, pp. 81–96, 2017.

[27] R. Balabin and E. Lomakina-Rumyantseva, "Neural network approach to quantum-chemistry data: Accurate prediction of density functional theory energies," *The Journal of chemical physics*, vol. 131, p. 074104, 09 2009.

[28] G. N, V. K, R. M A, and M. Palani, "Application of neural networks in diagnosing cancer disease using demographic data," *International Journal of Computer Applications*, vol. 1, 02 2010.

[29] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602 – 610, 2005. IJCNN 2005.

[30] A. Graves and J. Schmidhuber, "Offline arabic handwriting recognition with multidimensional recurrent neural networks," in *Advances in Neural Information Processing Systems 21 - Proceedings of the 2008 Conference*, pp. 545–552, 01 2008.

[31] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu, "Lstm network: a deep learning approach for short-term traffic forecast," *IET Intelligent Transport Systems*, vol. 11, no. 2, pp. 68–75, 2017.

[32] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *ArXiv*, 2016.

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Representations by Back-Propagating Errors*, p. 696–699. Cambridge, MA, USA: MIT Press, 1986.

[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[35] T. Ong, "Facebook's translations are now powered completely by ai." https://www.theverge.com/2017/8/4/16093872/facebook-ai-translations-artificial-intelligence. Accessed: 2020-02-10.

[36] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[37] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[38] K. Chellapilla, S. Puri, and P. Simard, "High Performance Convolutional Neural Networks for Document Processing," in *Tenth International Workshop on Frontiers in Handwriting Recognition* (G. Lorette, ed.), (La Baule (France)), Université de Rennes 1, Suvisoft, Oct. 2006. http://www.suvisoft.com.

[39] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[41] M. Coşkun, A. Uçar, Yildirim, and Y. Demir, "Face recognition based on convolutional neural network," in *2017 International Conference on Modern Electrical and Energy Systems (MEES)*, pp. 376–379, 2017.

[42] N. R. e. a. Esteva A, Kuprel B, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, 2017.

[43] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, (Cambridge, MA, USA), p. 2672–2680, MIT Press, 2014.

[44] M. Mustafa, D. Bard, W. Bhimji, Z. Lukić, R. Al-Rfou, and J. M. Kratochvil, "Cosmogan: creating high-fidelity weak lensing convergence maps using generative adversarial networks," *Computational Astrophysics and Cosmology*, vol. 6, May 2019.

[45] C. Wong, "The rise of ai supermodels." https://www.cdotrends.com/story/14300/rise-ai-supermodels. Accessed: 2020-06-21.

[46] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 5907–5915, 2017.

[47] C.-M. Tsai and Z.-M. Yeh, "Intelligent moving objects detection via adaptive frame differencing method," in *Intelligent Information and Database Systems* (A. Selamat, N. T. Nguyen, and H. Haron, eds.), (Berlin, Heidelberg), pp. 1–11, Springer Berlin Heidelberg, 2013.

[48] Y. Zheng and S. Peng, "Model based vehicle localization for urban traffic surveillance using image gradient based matching," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pp. 945–950, IEEE, 2012.

[49] K. SuganyaDevi, N. Malmurugan, and R. Sivakumar, "Efficient foreground extraction based on optical flow and smed for road traffic analysis," *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, vol. 1, no. 3, pp. 177–182, 2012.

[50] Y.-L. Chen, B.-F. Wu, H.-Y. Huang, and C.-J. Fan, "A real-time vision system for nighttime vehicle detection and traffic surveillance," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 5, pp. 2030–2044, 2010.

[51] V. Lempitsky and A. Zisserman, "Learning to count objects in images," in *Advances in neural information processing systems*, pp. 1324–1332, 2010.

[52] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 833–841, 2015.

[53] C. Arteta, V. Lempitsky, and A. Zisserman, "Counting in the wild," in *European conference on computer vision*, pp. 483–498, Springer, 2016.

[54] S. Zaraysky, *Language Is Music: Over 100 Fun & Easy Tips to Learn Foreign Languages*, vol. 3. Create Your World Books, 2009.

[55] E. Hajiramezanali, S. Z. Dadaneh, A. Karbalayghareh, M. Zhou, and X. Qian, "Bayesian multi-domain learning for cancer subtype discovery from next-generation sequencing count data," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, (Red Hook, NY, USA), p. 9133–9142, Curran Associates Inc., 2018.

[56] H. Oliveira, E. Ferreira, and J. A. dos Santos, "Truly generalizable radiograph segmentation with conditional domain adaptation," in *ArXiv*, 2019.

[57] Y. Zhang, Y. lin Wei, P. Zhao, S. Niu, Q. Wu, M. Tan, and J. Huang, "Collaborative unsupervised domain adaptation for medical image diagnosis," *ArXiv*, vol. abs/1911.07293, 2019.

[58] J. R. Finkel and C. D. Manning, "Hierarchical bayesian domain adaptation," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 602–610, 2009.

[59] Q. Li, "Literature survey: domain adaptation algorithms for natural language processing," *Department of Computer Science The Graduate Center, The City University of New York*, pp. 8–10, 2012.

[60] Y. Zhang, B. Tang, M. Jiang, J. Wang, and H. Xu, "Domain adaptation for semantic role labeling of clinical text," *Journal of the American Medical Informatics Association*, vol. 22, pp. 967–979, 06 2015.

[61] L. Duan, D. Xu, and S.-F. Chang, "Exploiting web images for event recognition in consumer videos: A multiple source domain adaptation approach," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1338–1345, IEEE, 2012.

[62] T. M. Mitchell, *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research, 1980.

[63] P. E. Utgoff, "Shift of bias for inductive concept learning," *Machine learning: An artificial intelligence approach*, vol. 2, pp. 107–148, 1986.

[64] L. Y. Pratt, "Discriminability-based transfer between neural networks," in *Advances in neural information processing systems*, pp. 204–211, 1993.

[65] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, pp. 41–75, Jul 1997.

[66] J. Baxter, "Theoretical models of learning to learn," in *Learning to Learn* (S. Thrun and L. Pratt, eds.), pp. 71–94, Springer US, 1998.

[67] J. Jiang and C. Zhai, "Instance weighting for domain adaptation in nlp," in *Proceedings of the 45th annual meeting of the association of computational linguistics*, pp. 264–271, 2007.

[68] L. Bruzzone and M. Marconcini, "Domain adaptation problems: A dasvm classification technique and a circular validation strategy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 770–787, May 2010.

[69] B. Gong, K. Grauman, and F. Sha, "Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation," in *Proceedings of the 30th International Conference on Machine Learning* (S. Dasgupta and D. McAllester,

eds.), vol. 28 of *Proceedings of Machine Learning Research*, (Atlanta, Georgia, USA), pp. 222–230, PMLR, 17–19 Jun 2013.

[70] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine Learning*, vol. 79, pp. 151–175, May 2010.

[71] G. Csurka, "A comprehensive survey on domain adaptation for visual applications," *Domain Adaptation in Computer Vision Applications*, pp. 1–35, 2017.

[72] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135 – 153, 2018.

[73] A. Iyer, S. Nath, and S. Sarawagi, "Maximum mean discrepancy for class ratio estimation: Convergence bounds and kernel selection," in *International Conference on Machine Learning*, pp. 530–538, 2014.

[74] Z. C. Lipton, Y.-X. Wang, and A. Smola, "Detecting and correcting for label shift with black box predictors," *arXiv preprint arXiv:1802.03916*, 2018.

[75] A. Storkey, "When training and test sets are different: characterizing learning transfer," *Dataset shift in machine learning*, pp. 3–28, 2009.

[76] M. Gong, K. Zhang, T. Liu, D. Tao, C. Glymour, and B. Schölkopf, "Domain adaptation with conditional transferable components," in *International Conference on Machine Learning*, pp. 2839–2848, 2016.

[77] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2200–2207, 2013.

[78] C. Cortes, Y. Mansour, and M. Mohri, "Learning bounds for importance weighting," in *Advances in Neural Information Processing Systems*, pp. 442–450, 2010.

[79] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of statistical planning and inference*, vol. 90, no. 2, pp. 227–244, 2000.

[80] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang, "Domain adaptation under target and conditional shift," in *International Conference on Machine Learning*, pp. 819–827, 2013.

[81] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent Data Analysis*, pp. 429–449, 2002.

[82] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: One-sided selection," in *In Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 179–186, Morgan Kaufmann, 1997.

[83] Y. Lin, Y. Lee, and G. Wahba, "Support vector machines for classification in nonstandard situations," *Machine Learning*, vol. 46, pp. 191–202, Jan 2002.

[84] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of Statistical Planning and Inference*, vol. 90, pp. 227–244, 10 2000.

[85] B. Zadrozny, "Learning and evaluating classifiers under sample selection bias," in *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, (New York, NY, USA), p. 114, Association for Computing Machinery, 2004.

[86] S. Bickel, M. Brückner, and T. Scheffer, "Discriminative learning for differing training and test distributions," in *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, (New York, NY, USA), p. 81–88, Association for Computing Machinery, 2007.

[87] V. N. Vapnik and A. Y. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," *Theory of Probability & Its Applications*, vol. 16, no. 2, pp. 264–280, 1971.

[88] J. Hoffman, M. Mohri, and N. Zhang, "Algorithms and theory for multiple-source adaptation," in *Advances in Neural Information Processing Systems 31* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 8246–8256, Curran Associates, Inc., 2018.

[89] J. Guo, D. Shah, and R. Barzilay, "Multi-source domain adaptation with mixture of experts," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, (Brussels, Belgium), pp. 4694–4703, Association for Computational Linguistics, Oct.-Nov. 2018.

[90] Y.-B. Kim, K. Stratos, and D. Kim, "Domain attention with an ensemble of experts," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Vancouver, Canada), pp. 643–653, Association for Computational Linguistics, July 2017.