UNIVERSIDADE DO PORTO

DOCTORAL THESIS

Learning from multi-entity data

Author:
Diogo PERNES

Supervisor:
Jaime S. CARDOSO

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy

at the

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência de Computadores

May 14, 2021

" I am and always will be the optimist, the hoper of far-flung hopes and the dreamer of improbable dreams "

Matt Smith as *The Doctor*, written by Matthew Graham

Acknowledgements

Acknowledge ALL the people!

UNIVERSIDADE DO PORTO

Abstract

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência de Computadores

Doctor of Philosophy

Learning from multi-entity data

by Diogo PERNES

This thesis is about something, I guess.

UNIVERSIDADE DO PORTO

Resumo

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência de Computadores

Doutoramento em Ciência de Computadores

Titulo da Tese em Português

por Diogo PERNES

Este tese é sobre alguma coisa

Contents

List of Figures

Notation and Conventions

In this section, we describe the notation adopted in this thesis. We mostly follow the notation proposed by Goodfellow et al. [1], which is also the recommended one by the International Conference on Learning Representations (ICLR).

This section must be reviewed after the thesis is complete. Some definitions might be missing and others are not even used, so those should be removed.

Numbers and Arrays

a	A scalar (integer or real)
$\boldsymbol{a}$	A vector
$\boldsymbol{A}$	A matrix
$\mathbf{A}$	A tensor
$\boldsymbol{I}_n$	Identity matrix with n rows and n columns
$\boldsymbol{I}$	Identity matrix with dimensionality implied by context
$\text{diag}(\boldsymbol{a})$	A square, diagonal matrix with diagonal entries given by $\boldsymbol{a}$
a	A scalar random variable
$\mathbf{a}$	A vector-valued random variable
$\mathbf{A}$	A matrix-valued random variable

Sets and Graphs

$\mathbb{R}$	The set of real numbers
$\{0, 1\}$	The set containing 0 and 1
$\{1, \dots, n\}$	The set of all integers between 1 and n
$[a, b]$	The real interval including a and b
$(a, b]$	The real interval excluding a but including b
$\mathcal{A} \backslash \mathcal{B}$	Set subtraction, i.e. the set containing the elements of $\mathcal{A}$ that are not in $\mathcal{B}$
$\mathcal{G}$	A graph
$\text{Pa}_{\mathcal{G}}(\text{x})$	The parents of node x in $\mathcal{G}$

Indexing

a_i	Element i of vector $\boldsymbol{a}$, with indexing starting at 1
$A_{i,j}$	Element i, j of matrix $\boldsymbol{A}$
a_i	Element i of the random vector $\mathbf{a}$

Linear Algebra Operations

$\boldsymbol{A}^\top$	Transpose of matrix $\boldsymbol{A}$
$\boldsymbol{A} \odot \boldsymbol{B}$	Element-wise (Hadamard) product of $\boldsymbol{A}$ and $\boldsymbol{B}$
$\det(\boldsymbol{A})$	Determinant of $\boldsymbol{A}$

Calculus

$\frac{dy}{dx}$	Derivative of y with respect to x
$\frac{\partial y}{\partial x}$	Partial derivative of y with respect to x
$\nabla_{\boldsymbol{x}} y$	Gradient of y with respect to $\boldsymbol{x}$
$\nabla_{\boldsymbol{X}} y$	Matrix derivatives of y with respect to $\boldsymbol{X}$
$\int f(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}$	Definite integral over the entire domain of $\boldsymbol{x}$
$\int_{\mathcal{X}} f(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}$	Definite integral with respect to $\boldsymbol{x}$ over the set $\mathcal{X}$

Probability and Information Theory

$\mathrm{a} \perp \mathrm{b}$	The random variables a and b are independent
$\mathrm{a} \perp \mathrm{b} \mid \mathrm{c}$	They are conditionally independent given c
$p(\mathrm{x})$	A probability distribution over the random variable x
$\mathrm{x} \sim p$	Random variable x has distribution p
$\mathrm{Supp}(\mathrm{x})$	Support of the random variable x
$\mathbb{E}_{\mathrm{x} \sim p}[f(\mathrm{x})]$	Expectation of $f(\mathrm{x})$ with respect to $p(\mathrm{x})$
$H(\mathrm{x})$	Shannon entropy of the random variable x
$D_{\mathrm{KL}}(p \Vert q)$	Kullback-Leibler divergence of p and q
$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution over $\mathbf{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

Functions

$f : \mathcal{A} \to \mathcal{B}$	The function f with domain $\mathcal{A}$ and range $\mathcal{B}$
$f \circ g$	Composition of the functions f and g
$f(\boldsymbol{x}; \boldsymbol{\theta})$	A function of $\boldsymbol{x}$ parametrized by $\boldsymbol{\theta}$. (Sometimes we write $f(\boldsymbol{x})$ and omit the argument $\boldsymbol{\theta}$ to lighten notation)
$\log x$	Natural logarithm of x
$\sigma(x)$	Logistic sigmoid, $\frac{1}{1+\exp(-x)}$
$\|\|\boldsymbol{x}\|\|_p$	L^p norm of vector $\boldsymbol{x}$
$\|\|\boldsymbol{x}\|\|$	L^2 norm of vector $\boldsymbol{x}$
$\|\|\boldsymbol{X}\|\|$	Frobenius norm of matrix $\boldsymbol{X}$
x^+	Positive part of x, i.e. $\max(0, x)$
$\mathbf{1}_{\text{condition}}$	is 1 if the condition is true, 0 otherwise

Sometimes we use a function f whose argument is a scalar but apply it to a vector, matrix, or tensor: $f(\boldsymbol{x})$, $f(\boldsymbol{X})$, or $f(\mathbf{X})$. This denotes the application of f to the array element-wise. For example, if $\mathbf{C} = \sigma(\mathbf{X})$, then $C_{i,j,k} = \sigma(X_{i,j,k})$ for all valid values of i, j and k.

Chapter 1

Background

1.1 Introduction

We will use deep neural networks and probabilistic models extensively throughout this thesis. Although the basic concepts of the former should be fairly familiar to most readers, the latter might be less widely known. Moreover, some of the tools and algorithms that we are going to use are not so trivial and therefore they should better be introduced first, for the sake of completeness and readability.

This chapter will then focus on providing a brief yet rigorous background on the aforementioned subject. We shall start by clarifying some notations and conventions we shall adopt (Section 1.2). Then we proceed with a short introduction to Bayesian networks, motivated by the factorization properties of joint probability functions (Section **??**). Expectation-Maximization (EM) is presented as an efficient algorithm to learn probabilistic models with unobserved variables (Section **??**). We then review the hidden Markov model (HMM), which will play a central role in Chapter 2, and present the instantiation of the EM algorithm for this particular model (Section **??**). The chapter is concluded with a brief introduction to variational inference (Section **??**). Under this setting, the variational autoencoder will deserve special attention (Section **??**), as it will be one of the models employed in Chapter 4.

1.2 Useful definitions and conventions

In this section, we introduce some further notations and definitions that will be used throughout this document.

It is important to remark that the same notation is used to denote discrete and continuous random variables as well as to denote probability mass functions and probability density functions. Specifically, given a random variable x taking values in the set $\mathrm{Val}(\mathrm{x})$, $p(\mathrm{x})$ denotes the probability mass of x, if $\mathrm{Val}(\mathrm{x})$ is discrete, or the probability density of x, otherwise. In either case, the *support* of $p(\mathrm{x})$ is defined as $\mathrm{Supp}(p(\mathrm{x})) \triangleq \{x \in \mathrm{Val}(\mathrm{x}) : p(\mathrm{x} = x) > 0\}$. Moreover, when we want to denote the probability (density) of some arbitrary but fixed value $x \in \mathrm{Val}(\mathrm{x})$, we often use $p(x)$ as a short for $p(\mathrm{x} = x)$.

The joint probability function of x and y is denoted by $p(\mathrm{x}, \mathrm{y})$ and the corresponding conditionals of y given x and x given y are denoted by the usual $p(\mathrm{y} \mid \mathrm{x})$ and $p(\mathrm{x} \mid \mathrm{y})$, respectively. Again, x and y can be both discrete, both continuous, or one continuous and the other discrete. For three or more random variables the notation generalizes naturally.

The marginalization of $p(\mathrm{x}, \mathrm{y})$ with respect to a discrete y is written as:

$$\sum_{\mathrm{y}} p(\mathrm{x}, \mathrm{y}) \triangleq \sum_{y \in \mathrm{Val}(\mathrm{y})} p(\mathrm{x}, y) = p(\mathrm{x}), \tag{1.1}$$

and the marginalization of $p(\mathrm{x}, \mathrm{y})$ with respect to a continuous x is written as:

$$\int p(\mathrm{x}, \mathrm{y}) \, \mathrm{d}\mathrm{x} \triangleq \int_{\mathrm{Val}(\mathrm{x})} p(x, \mathrm{y}) \, \mathrm{d}x = p(\mathrm{y}). \tag{1.2}$$

Note that for brevity we omit the domain of the summation or integration, as this is defined implicitly by the set where the random variable is defined. The integral notation is also used for the marginalization with respect to random variables whose type is unspecified. Similarly, if $f(\mathrm{x})$ is a function of a random variable x,

$$\sum_{\mathrm{x}} f(\mathrm{x}) \triangleq \sum_{x \in \mathrm{Val}(\mathrm{x})} f(x) \quad \text{or} \quad \int f(\mathrm{x}) \, \mathrm{d}\mathrm{x} \triangleq \int_{\mathrm{Val}(\mathrm{x})} f(x) \, \mathrm{d}x, \tag{1.3}$$

depending on whether x is discrete or continuous, respectively. Hence, we can write the *expectation* of $f(\mathrm{x})$ with respect to $p(\mathrm{x})$ as:

$$\mathbb{E}_{\mathrm{x} \sim p(\mathrm{x})} \left[f(\mathrm{x}) \right] \triangleq \sum_{\mathrm{x}} f(\mathrm{x}) p(\mathrm{x}) \quad \text{or} \quad \mathbb{E}_{\mathrm{x} \sim p(\mathrm{x})} \left[f(\mathrm{x}) \right] \triangleq \int f(\mathrm{x}) p(\mathrm{x}) \, \mathrm{d}\mathrm{x}, \tag{1.4}$$

for a discrete or continuous x, respectively.

1.3 Bayesian networks

1.3.1 Definition and structural properties

Given m random variables $x_1, x_2, \ldots, x_m$, the *chain rule of probability* allows the factorization of their joint distribution as a product of conditional distributions:*

$$p(x^{(1)}, x^{(2)}, \ldots, x^{(m)}) = p(x^{(1)}) \prod_{j=1}^{m} p(x^{(j)} \mid x^{(1)}, x^{(2)}, \ldots, x^{(j-1)}). \tag{1.5}$$

Starting from a factorization of a joint distribution into conditionals, we may build a directed graph $\mathcal{G}$ with m vertices, one for each random variable, where there exists an edge $i \to j$ if and only if there is a factor where $x^{(j)}$ is conditioned on $x^{(i)}$. Such a graph is known as a *Bayesian network*. From its definition, we see that the factorization in equation (1.5) corresponds to the graph in Figure 1.1. Clearly, a Bayesian network defines a bijection between random variables and graph nodes, so with a slight abuse of terminology we represent and refer to nodes by the random variable they are associated with, rather than by their index.

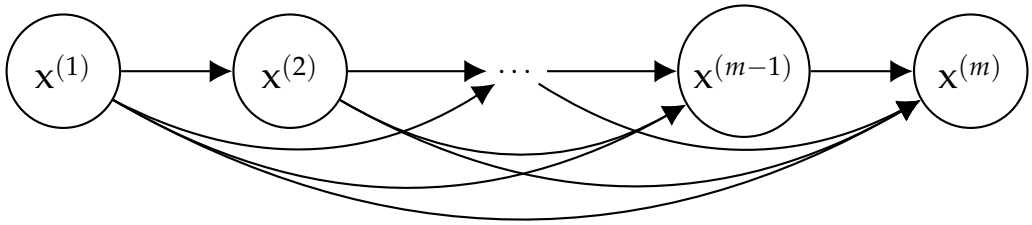


FIGURE 1.1: A complete Bayesian network.

Since the factorization in equation (1.5) is general (i.e. it does not assume any conditional independencies between random variables), the Bayesian network in Figure 1.1 is said to be *complete*. When conditional independencies are present, the graph becomes sparser. For instance, if, for all $j \geq 3$, $x^{(j)}$ is conditionally independent of $x^{(1)}, x^{(2)}, \ldots, x^{(j-2)}$ given $x^{(j-1)}$, then $p(x^{(j)} \mid x^{(1)}, x^{(2)}, \ldots, x^{(j-1)}) = p(x^{(j)} \mid x^{(j-1)})$ and thus the Bayesian network reduces to the *chain* represented in Figure 1.2. A sparser graph implies a more compact parametrization of the model. For instance, if each of the m random variables takes values on a discrete set of size s, defining the joint distribution corresponding to a complete Bayesian network (Figure 1.1) would require $s^m - 1$ parameters. However, in the same setting, the joint distribution corresponding to a chain (Figure 1.2) can be fully

*The expression on the right-hand side of equation (1.5) is not well defined outside the support of $p(x^{(1)}), p(x^{(1)}, x^{(2)}), \ldots, p(x^{(1)}, x^{(2)}, \ldots, x^{(m-1)})$. For those values, one has $p(x^{(1)}, x^{(2)}, \ldots, x^{(m)}) = 0$.

described using table conditional probability distributions (CPDs) with less than s^2m parameters in total.

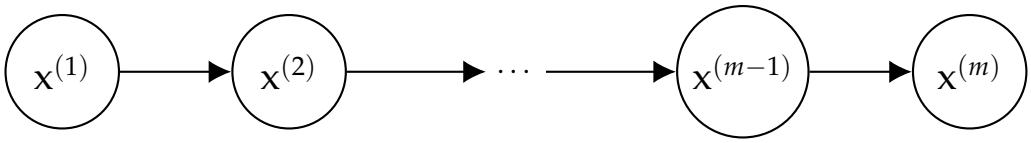


FIGURE 1.2: A chain.

An important property of Bayesian networks is the fact that they are always directed *acyclic* graphs. The non-existence of cycles allows us to recover the factorization of the joint distribution by examining the structure of the graph $\mathcal{G}$ and applying the formula:

$$p(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}) = \prod_{j=1}^{m} p(\mathbf{x}^{(j)} \mid \mathrm{Pa}_{\mathcal{G}}(\mathbf{x}^{(j)})), \tag{1.6}$$

where $\mathrm{Pa}_{\mathcal{G}}(\mathbf{x}^{(j)})$ are the parents of node $\mathbf{x}^{(j)}$ in $\mathcal{G}$. It is important to highlight that, although there is a one-to-one correspondence between a factorization of a joint distribution into conditional distributions and a Bayesian network, the same joint distribution can sometimes be factorized in multiple different but equivalent forms, each one corresponding to a different Bayesian network. Note, for instance, that equation (1.5) is just one out of $m!$ possible ways of factorizing $p(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)})$ using the chain rule and any of those factorizations would yield a different graph. This is also the case for chains and *forks*, which we discuss next.

1.3.2 Chains and forks

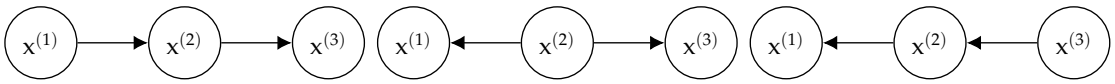
Let us consider the case where we have three random variables $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}$ and assume that $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(3)}$ are independent given $\mathbf{x}^{(2)}$. Under this setting, we have:

$$p(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}) = p(\mathbf{x}^{(1)}) p(\mathbf{x}^{(2)} \mid \mathbf{x}^{(1)}) p(\mathbf{x}^{(3)} \mid \mathbf{x}^{(2)}) \tag{1.7}$$

$$= p(\mathbf{x}^{(2)}) p(\mathbf{x}^{(1)} \mid \mathbf{x}^{(2)}) p(\mathbf{x}^{(3)} \mid \mathbf{x}^{(2)}) \tag{1.8}$$

$$= p(\mathbf{x}^{(3)}) p(\mathbf{x}^{(2)} \mid \mathbf{x}^{(3)}) p(\mathbf{x}^{(1)} \mid \mathbf{x}^{(2)}) \tag{1.9}$$

The three equivalent factorizations (1.7) – (1.9) correspond to the three Bayesian networks in Figure 1.3, respectively from left to right.



(A) A three-variable chain. (B) A fork. (C) A three-variable reversed chain.

FIGURE 1.3: Equivalent three-variable Bayesian networks.

1.3.3 Immoralities

Now let us assume that $x^{(1)}$ and $x^{(3)}$ are marginally independent, which notably does not imply that they are conditionally independent given $x^{(2)}$. In this case, the joint distribution factorizes as:

$$p(x^{(1)}, x^{(2)}, x^{(3)}) = p(x^{(1)})p(x^{(3)})p(x^{(2)} \mid x^{(1)}, x^{(3)}) \tag{1.10}$$

and this factorization is unique in the sense that no other implies the same set of conditional independencies. The corresponding Bayesian network is represented in Figure 1.4. A subgraph consisting of three nodes where two of them are not connected and the other one is a child of both is an *immorality*. The node where the two incoming edges coincide is a *collider*. Thus, in the present example, $x^{(1)}$, $x^{(2)}$, and $x^{(3)}$ form an immorality where $x^{(2)}$ is the collider.

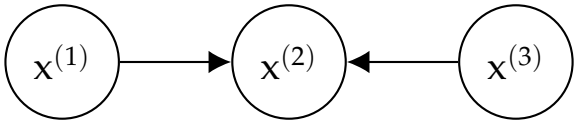


FIGURE 1.4: An immorality.

1.3.4 D-separation

Armed with the notions of chains, forks, and immoralities, and the conditional independencies implied by them, we are ready to introduce the concept of *blocked paths* and *d-separation*. The latter provides a powerful graphical method to discover all the conditional independencies implied by any Bayesian network.

Specifically, we say that an undirected path between two nodes x and y is *blocked* by a (potentially empty) conditioning set $\mathcal{S}$ if at least one of the following two conditions holds:

- Along the path there is a chain or a fork which includes at least one node in $\mathcal{S}$.

- Along the path there is a collider and neither the collider nor any of its descendants are in $\mathcal{S}$.

The two nodes x and y are *d-separated* by a set of nodes $\mathcal{S}$ if conditioning on $\mathcal{S}$ blocks all paths between x and y. Remarkably, if x and y are d-separated by $\mathcal{S}$, then they are conditionally independent given $\mathcal{S}$ (see Koller and Friedman [2] for a proof).

1.3.5 The hidden Markov model

A classical example of a Bayesian network is the hidden Markov model, represented in Figure 1.5. The HMM will be the backbone of both models presented in Chapter 2 and hence it is appropriate to introduce it here.

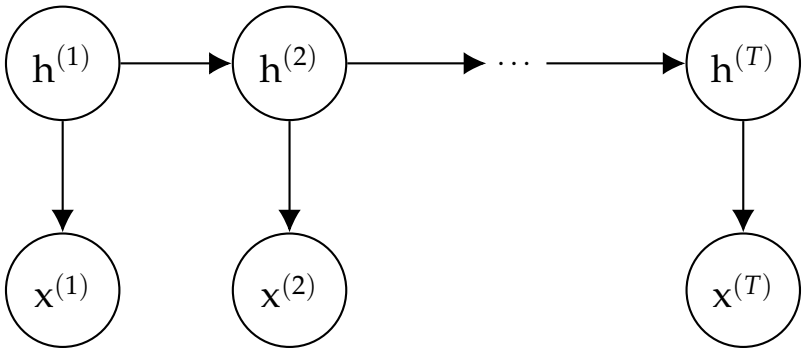


FIGURE 1.5: Hidden Markov model represented as Bayesian network.

The structure of the graph in Figure 1.5 corresponds to the following factorization:

$$p(\mathrm{x}^{(1)},\mathrm{x}^{(2)},\ldots,\mathrm{x}^{(T)},\mathrm{h}^{(1)},\mathrm{h}^{(2)},\ldots,\mathrm{h}^{(t)}) = p(\mathrm{h}^{(1)})p(x^{(1)} \mid h^{(1)})\prod_{t=2}^{T} p(h^{(t)} \mid h^{(t-1)})p(x^{(t)} \mid h^{(t)}) \tag{1.11}$$

Here, $\mathrm{x}^{(1)},\mathrm{x}^{(2)},\ldots,\mathrm{x}^{(T)}$ is the sequence of *observations* and $\mathrm{h}^{(1)},\mathrm{h}^{(2)},\ldots,\mathrm{h}^{(T)}$ is the sequence of *hidden states*.* Observations can be either discrete or continuous, while hidden states are usually discrete, although continuous state versions exist (Turin [3]).

By analyzing d-separation in Figure 1.5, two fundamental assumptions of the HMM are revealed:

- For all t, the hidden state $\mathrm{h}^{(t)}$ is conditionally independent of all past hidden states $\mathrm{h}^{(1)},\mathrm{h}^{(2)},\ldots,\mathrm{h}^{(t-2)}$ given the previous hidden state $\mathrm{h}^{(t-1)}$ — *Markov* assumption.
- For all t, the observation $\mathrm{x}^{(t)}$ is conditionally independent of all past and future observations $\mathrm{x}^{(1)},\ldots,\mathrm{x}^{(t-1)},\mathrm{x}^{(t+1)},\ldots,\mathrm{x}^{(T)}$ and of all past and future hidden states

*The word *hidden* refers to the fact that the states are usually not observed in the training data.

$\mathrm{h}^{(1)}, \ldots, \mathrm{h}^{(t-1)}, \mathrm{h}^{(t+1)}, \ldots, \mathrm{h}^{(T)}$ given the current hidden state $\mathrm{h}^{(t)}$ — *output independence* assumption.

Furthermore, it is also assumed that both the state transition distribution $p(\mathrm{h}^{(t)} \mid \mathrm{h}^{(t-1)})$ and the emission distribution $p(\mathrm{x}^{(t)} \mid \mathrm{h}^{(t)})$ are *stationary*, i.e. they are the same for all t. Besides reducing the number of parameters required to describe the model, the stationarity assumption allows the model to work with sequences of arbitrary length T. It should be noted that, in some applications, these assumptions are relaxed yielding HMMs where the hidden state depends on the k previous hidden states (k-th order HMM) or where the distributions are time-dependent (non-stationary HMM).

The independence assumptions of the HMM also have the advantage of making inference in this model a tractable problem. Here, we will be particularly interested in computing $p(\mathrm{x}^{(1)}, \mathrm{x}^{(2)}, \ldots, \mathrm{x}^{(T)})$, $p(\mathrm{h}^{(t)} \mid \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathrm{x}^{(T)})$, and $p(\mathrm{h}^{(t-1)}, \mathrm{h}^{(t)} \mid \mathrm{x}^{(1)}, \mathrm{x}^{(2)}, \ldots, \mathrm{x}^{(T)})$, which can be obtained with Algorithm 1.1 (see Bishop [4] for the derivation).

Algorithm 1.1 Forward-backward algorithm (Rabiner and Juang [5]).

1: **Inputs:** An observation sequence $x^{(1)}, x^{(2)}, \ldots, x^{(T)}$ and an HMM with defined initial state, state transition, and emission distributions.

2: Initialize $\alpha_h(1) \coloneqq p(\mathrm{h}^{(1)} = h) p(x^{(1)} \mid \mathrm{h}^{(1)} = h)$ and $\beta_h(T) \coloneqq 1$ for all h in the state dictionary.

3: Compute the forward recursion $\alpha_h(t) \coloneqq p(x^{(t)} \mid \mathrm{h}^{(t)} = h) \sum_{\mathrm{h}^{(t-1)}} \alpha_{\mathrm{h}^{(t-1)}}(t-1) p(\mathrm{h}^{(t)} = h \mid \mathrm{h}^{(t-1)})$ for all h in the state dictionary and $t = 2, 3, \ldots, T$.

4: Compute the backward recursion $\beta_h(t) \coloneqq p(x^{(t+1)} \mid \mathrm{h}^{(t+1)} = h) \sum_{\mathrm{h}^{(t+1)}} \beta_{\mathrm{h}^{(t+1)}}(t+1) p(\mathrm{h}^{(t+1)} \mid \mathrm{h}^{(t)} = h)$ for all h in the state dictionary and $t = T-1, T-2, \ldots, 1$.

5: Obtain the marginal likelihood of the observed sequence as $p(x^{(1)}, x^{(2)}, \ldots, x^{(T)}) \coloneqq \sum_{\mathrm{h}^{(T)}} \alpha_{\mathrm{h}^{(T)}}(T)$.

6: Obtain the state posterior distribution as $p(\mathrm{h}^{(t)} = h \mid x^{(1)}, x^{(2)}, \ldots, x^{(T)}) = \frac{\alpha_h(t)\beta_h(t)}{p(x^{(1)}, x^{(2)}, \ldots, x^{(T)})}$ for all h in the state dictionary.

7: Obtain the joint posterior distribution of two consecutive states as $p(\mathrm{h}^{(t-1)} = h, \mathrm{h}^{(t)} = h' \mid x^{(1)}, x^{(2)}, \ldots, x^{(T)}) = \frac{\alpha_h(t-1) p(\mathrm{h}^{(t)} = h' \mid \mathrm{h}^{(t-1)} = h) p(x^{(t)} \mid \mathrm{h}^{(t)} = h') \beta_{h'}(t)}{p(x^{(1)}, x^{(2)}, \ldots, x^{(T)})}$ for all h, h' in the state dictionary.

1.4 The Expectation-Maximization algorithm

1.4.1 General formulation

Given a dataset $\{\boldsymbol{x}_i\}_{i=1}^n$ and a probabilistic model with m random variables, the parameters Θ of the model are often estimated using the *maximum likelihood criterion*, i.e. the optimal parameters maximize:

$$J(\Theta) \triangleq \sum_{i=1}^{n} \log p(\boldsymbol{x}_i; \Theta). \tag{1.12}$$

When every example $\boldsymbol{x}_i$ is a vector containing an observation for each of the m random variables in the model and p is a usual family of distributions (e.g. exponential family or table CPD), objective (1.12) is concave and therefore it has a unique maximizer Θ^*. For instance, when Θ corresponds to the parameters of table CPDs, Θ^* is given by the empirical conditional probabilities obtained from the provided dataset (see Koller and Friedman [2] for a proof).

However, if there exist $\boldsymbol{x}_i$ which do not contain observations for some of the m random variables, the maximum likelihood objective becomes non-concave and admits multiple local optima. A procedure to find one of those optimum values is (stochastic) gradient ascent over the likelihood function. A usually better alternative is the Expectation-Maximization algorithm, explained next, since it exhibits faster convergence.

For simplicity, assume that if a random variable is observed in a given example $\boldsymbol{x}_i$ then it is observed in all remaining $n-1$ examples as well, i.e. assume that the m random variables can be partitioned as $\{\mathrm{x}^{(j)}\}_{j=1}^{o} \cup \{\mathrm{z}^{(j)}\}_{j=1}^{l}$ where the $\mathrm{x}^{(j)}$ are the o observed variables and the $\mathrm{z}^{(j)}$ are the l unobserved (or *latent*) variables and $o + l = m$.* Let $\mathbf{x} \triangleq \left(\mathrm{x}^{(1)}, \ldots, \mathrm{x}^{(o)}\right)$ and $\mathbf{z} \triangleq \left(\mathrm{z}^{(1)}, \ldots, \mathrm{z}^{(l)}\right)$ be the vectors of observed and latent variables, respectively. Under this setting, objective (1.12) becomes:

$$J(\Theta) = \sum_{i=1}^{n} \log \int p(\boldsymbol{x}_i, \mathbf{z}; \Theta)\, \mathrm{d}\mathbf{z}. \tag{1.13}$$

Now, let $q(\mathrm{z})$ be an arbitrary distribution such that $\mathrm{Supp}(q(\mathbf{z})) \supseteq \mathrm{Supp}(p(\mathbf{z} \mid \boldsymbol{x}_i; \Theta))$ for all i. Then,

$$J(\Theta) = \sum_{i=1}^{n} \log \int \frac{p(\boldsymbol{x}_i, \mathbf{z}; \Theta)}{q(\mathbf{z})} q(\mathbf{z})\, \mathrm{d}\mathbf{z}$$

*This assumption is not necessary for the EM algorithm to be valid. However, it makes the exposition simpler and it will hold for all latent variable models considered in this thesis.

$$= \sum_{i=1}^{n} \log \mathbb{E}_{\mathbf{z}\sim q}\left[\frac{p(\boldsymbol{x}_i, \mathbf{z}; \Theta)}{q(\mathbf{z})}\right]$$
$$\geq \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z}\sim q}\left[\log \frac{p(\boldsymbol{x}_i, \mathbf{z}; \Theta)}{q(\mathbf{z})}\right] \triangleq \mathrm{ELBO}(q, \Theta), \tag{1.14}$$

where the inequality is a particular case of Jensen's inequality. The EM algorithm proceeds maximizes this *evidence lower bound* (ELBO) in the following block coordinate ascent manner, until a convergence criterion is satisfied:

- Keep $\Theta = \Theta^{(-)}$ (fixed) and find $q^* = \arg\max_q \mathrm{ELBO}(q, \Theta^{(-)})$ — *E-step*.
- Keep $q = q^{(-)}$ (fixed) and find $\Theta^* = \arg\max_\Theta \mathrm{ELBO}(q^{(-)}, \Theta)$ — *M-step*.

1.4.2 E-step

Maximizing the ELBO with respect to q is obviously equivalent to the minimization of the difference $J(\Theta^{(-)}) - \mathrm{ELBO}(q, \Theta^{(-)})$ with respect to the same distribution. It is easy to verify that:

$$J(\Theta^{(-)}) - \mathrm{ELBO}(q, \Theta^{(-)}) = \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z}\sim q}\left[\log \frac{q(\mathbf{z})}{p(\mathbf{z} \mid \boldsymbol{x}_i; \Theta^{(-)})}\right] = D_{\mathrm{KL}}(q(\mathbf{z}) \| p(\mathbf{z} \mid \boldsymbol{x}_i; \Theta^{(-)})), \tag{1.15}$$

where $D_{\mathrm{KL}}(q(\mathbf{z}) \| p(\mathbf{z} \mid \boldsymbol{x}_i; \Theta^{(-)}))$ denotes the Kullback-Leibler divergence between the distributions $q(\mathbf{z})$ and $p(\mathbf{z} \mid \boldsymbol{x}_i; \Theta^{(-)})$. This quantity is always non-negative, being zero if and only if the two distributions coincide. Thus, the optimal q is $q^*(\mathbf{z}) = p(\mathbf{z} \mid \boldsymbol{x}_i; \Theta^{(-)})$.

1.4.3 M-step

Having found the optimal q, we now maximize the ELBO with respect to the parameters Θ. Plugging in $q^{(-)}(z) = p(\mathbf{z} \mid \boldsymbol{x}_i; \Theta^{(-)})$ yields:

$$\mathrm{ELBO}(q^{(-)}, \Theta) = \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\boldsymbol{x}_i;\Theta^{(-)})}\left[\log p(\boldsymbol{x}_i, \mathbf{z}; \Theta)\right] - \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\boldsymbol{x}_i;\Theta^{(-)})}\left[\log p(\mathbf{z} \mid \boldsymbol{x}_i; \Theta^{(-)})\right], \tag{1.16}$$

where the second term is constant with respect to Θ and therefore can be discarded. The first term is often concave and the maximizer Θ^* can be written in closed form. This is for instance the case for table CPDs, whose closed-form EM update equations are described next.

1.4.4 EM for table CPDs

We now provide the E-step and M-step update equations applicable to any Bayesian network $\mathcal{G}$ whose conditional distributions are described by table CPDs. The derivation of these formulas can be found in Koller and Friedman [2].

E-step: At this stage, using the parameters $\Theta^{(-)}$ obtained at the previous M-step (or the initial parameters if at the beginning), we should obtain the *expected sufficient statistics*. For this purpose, for each random variable w in the Bayesian network $\mathcal{G}$ and for each $(w, \boldsymbol{v}) \in \mathrm{Val}(\mathrm{w}, \mathrm{Pa}_{\mathcal{G}}(\mathrm{w}))$:

- Compute the posterior distributions $p(\mathrm{w}, \mathrm{Pa}_{\mathcal{G}}(\mathrm{w}) \mid \boldsymbol{x}_i; \Theta^{(-)})$ and $p(\mathrm{Pa}_{\mathcal{G}}(\mathrm{w}) \mid \boldsymbol{x}_i; \Theta^{(-)})$ for each i.
- Compute the expected sufficient statistics as:

$$M(w, \boldsymbol{v}) \triangleq \sum_{i=1}^{n} p(w, \boldsymbol{v} \mid \boldsymbol{x}_i; \Theta^{(-)}), \tag{1.17}$$

$$M(\boldsymbol{v}) \triangleq \sum_{i=1}^{n} p(\boldsymbol{v} \mid \boldsymbol{x}_i; \Theta^{(-)}). \tag{1.18}$$

M-step: Now, given the expected sufficient statistics computed at the previous E-step, we update the parameters Θ using:

$$\theta_{w|\boldsymbol{v}} = \frac{M(w, \boldsymbol{v})}{M(\boldsymbol{v})}, \tag{1.19}$$

where $\theta_{w|\boldsymbol{v}}$ denotes the table entry corresponding to the probability $p(\mathrm{w} = w \mid \mathrm{Pa}_{\mathcal{G}}(\mathrm{w}) = \boldsymbol{v})$. The set Θ is the union of all these parameters.

1.4.5 EM for the discrete emission HMM

An example of a table CPD model with latent variables is the HMM with discrete emission distribution. Under this setting, the hidden state $\mathrm{h}^{(t)}$ and the observation $\mathrm{x}^{(t)}$ both take values in discrete and finite sets. Thus, we can use equations (1.17) and (1.18) together with the structure of the Bayesian network in Figure 1.5 and Algorithm 1.1 to obtain the EM training procedure for this model, which is summarized in Algorithm 1.2.

The parameters to be learned here are:

- the initial state probabilities $\pi_h \triangleq p(\mathrm{h}^{(1)} = h)$, for all h in the state dictionary;

- the state transition probabilities $A_{h,h'} \triangleq p(\mathrm{h}^{(t)} = h' \mid \mathrm{h}^{(t-1)} = h)$, for all h, h' in the state dictionary;
- the emission probabilities $B_{h,x} \triangleq p(\mathrm{x}^{(t)} = x \mid \mathrm{h}^{(t)} = h)$, for all h and x in the state and observation dictionaries, respectively.

Algorithm 1.2 Baum-Welch algorithm (Baum [6]).

Inputs: A training set $\{\boldsymbol{x}_i\}_{i=1}^n$ of observation sequences $x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(T_i)}$, a set of initial parameters $\Theta^{(0)}$ for the HMM, and the number of training iterations `trainIter`.

for $j = 1, \ldots,$ `trainIter` **do**

E-step:

Obtain the state posteriors $\gamma_{i,h}(t) := p(\mathrm{h}^{(t)} = h \mid \boldsymbol{x}_i; \Theta^{(j-1)})$ and $\xi_{i,h,h'}(t) := p(\mathrm{h}^{(t-1)} = h, \mathrm{h}^{(t)} = h' \mid \boldsymbol{x}_i; \Theta^{(j-1)})$ for $i = 1, \ldots, n$, $t = 1, \ldots, T_i$, and h, h' in the state dictionary, as done in Algorithm 1.1.

M-step:

$\pi_h := \frac{1}{n} \sum_{i=1}^n \gamma_{i,h}(1)$, for all h in the state dictionary;

$A_{h,h'} := \frac{\sum_{i=1}^n \sum_{t=1}^{T_i} \xi_{i,h,h'}(t)}{\sum_{i=1}^n \sum_{t=1}^{T_i} \gamma_{i,h}(t)}$, for all h, h' in the state dictionary;

$B_{h,x} := \frac{\sum_{i=1}^n \sum_{t=1}^{T_i} \gamma_{i,h}(t) \mathbf{1}_{x_i^{(t)} = x}}{\sum_{i=1}^n \sum_{t=1}^{T_i} \gamma_{i,h}(t)}$, for all h and x in the state and observation dictionaries, respectively;

$\Theta^{(j)} := \bigcup_{h,h',x} \left\{ \pi_h, A_{h,h'}, B_{h,x} \right\}$.

end for

1.5 Variational autoencoder

1.5.1 Formulation

In Section 1.4.2, we have shown that the posterior distribution $p(\mathbf{z} \mid \mathbf{x}; \Theta)$ was the optimal solution for the maximization of the ELBO with respect to the *variational* distribution $q(\mathbf{z})$. In the general case, this posterior is given by:

$$p(\mathbf{z} \mid \mathbf{x}; \Theta) = \frac{p(\mathbf{x}, \mathbf{z}; \Theta)}{\int p(\mathbf{x}, \mathbf{z}; \Theta) \, \mathrm{d}\mathbf{z}}, \tag{1.20}$$

where the integral in the denominator is usually designated as the *partition function*. When $\mathbf{z}$ is high-dimensional, computing the partition function becomes intractable and, therefore, the computation of the exact posterior is infeasible. This is where variational methods come into play.

Variational methods are used as approximation methods in a wide variety of settings, from quantum mechanics (Sakurai and Napolitano [7]) to statistics (Rustagi [8]). In the context of statistics and Bayesian inference, variational methods are used not only to find an approximate posterior in the E-step of the EM algorithm, but also as a general approximate inference tool.

Suppose now that we want to learn a conditional distribution $p(\mathbf{X} \mid \mathbf{z}; \boldsymbol{\theta}_e)$ of images $\mathbf{X}$ conditioned on latent codes $\mathbf{z}$ by implementing a neural network with parameters $\boldsymbol{\theta}_d$. This distribution combined with a prior $p(\mathbf{z})$ over the latent codes define the image likelihood $p(\mathbf{X}; \boldsymbol{\theta}_d)$ as:

$$p(\mathbf{X}; \boldsymbol{\theta}_d) = \int p(\mathbf{X} \mid \mathbf{z}; \boldsymbol{\theta}_d) p(\mathbf{z}) \, \mathrm{d}\mathbf{z}. \tag{1.21}$$

Therefore, in principle, this model can be trained using the maximum likelihood criterion and the EM algorithm. Note, however, that equation (1.21) is the partition function of this model. The issue here is the marginalization over $\mathbf{z}$, which is intractable since one of the integrands is a neural network and $\mathbf{z}$ is usually high-dimensional. To overcome this issue, a parametric variational distribution $q(\mathbf{z} \mid \mathbf{X}; \boldsymbol{\theta}_e)$ is introduced as a replacement for the true posterior $p(\mathbf{z} \mid \mathbf{X}; \boldsymbol{\theta}_d)$. This variational distribution is itself implemented with a deep neural network. If the family of distributions parametrized by the network is rich enough, it will be able to provide a tight approximation of the true posterior and hence maximize the ELBO. Formally, the objective is:

$$\max_{\boldsymbol{\theta}_e, \boldsymbol{\theta}_d} \left\{ \mathrm{ELBO}(\boldsymbol{\theta}_e, \boldsymbol{\theta}_d) = \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} \mid \boldsymbol{X}_i; \boldsymbol{\theta}_e)} \left[\log \frac{p(\boldsymbol{X}_i \mid \mathbf{z}; \boldsymbol{\theta}_d) p(\mathbf{z})}{q(\mathbf{z} \mid \boldsymbol{X}_i; \boldsymbol{\theta}_e)} \right] \right\}. \tag{1.22}$$

Note that the network $q(\mathrm{z} \mid \mathbf{X}; \boldsymbol{\theta}_e)$ defines a stochastic mapping from input images to latent codes and the network $p(\mathbf{X} \mid \mathbf{z}; \boldsymbol{\theta}_d)$ defines its inverse mapping. Hence, the former is an *encoder*, the latter is a *decoder*, and the model itself is called a *variational autoencoder* (VAE, Kingma and Welling [9]). Regarding the choice of prior $p(\mathbf{z})$ and approximate

posterior $q(\mathbf{z} \mid \mathbf{X}; \boldsymbol{\theta}_e)$, both are often set as Gaussian, specifically:

$$p(\mathbf{z}) \triangleq \mathcal{N}(\mathbf{z}; 0, \boldsymbol{I}), \tag{1.23}$$

$$q(\mathbf{z} \mid \boldsymbol{X}; \boldsymbol{\theta}_e) \triangleq \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_e(\boldsymbol{X}; \boldsymbol{\theta}_e), \mathrm{diag}(\boldsymbol{\sigma}_e(\boldsymbol{X}; \boldsymbol{\theta}_e))^2), \tag{1.24}$$

where, in the latter, $\boldsymbol{\mu}_e(\boldsymbol{X}; \boldsymbol{\theta}_e)$ and $\boldsymbol{\sigma}_e^2(\boldsymbol{X}; \boldsymbol{\theta}_e)$ are the two outputs of the encoder network. The conditional likelihood $p(\mathbf{X} \mid \mathbf{z}; \boldsymbol{\theta}_d)$ is usually a Bernoulli distribution, for black and white images, or a Gaussian, for grayscale and color images. In the latter case,

$$p(\mathbf{X} \mid \boldsymbol{z}; \boldsymbol{\theta}_d) \triangleq \mathcal{N}(\mathbf{X}; \boldsymbol{\mu}_d(\boldsymbol{z}; \boldsymbol{\theta}_d), \sigma_d^2 \boldsymbol{I}), \tag{1.25}$$

where $\sigma_d^2 > 0$ is a hyperparameter. Note that, despite all distributions are simple Gaussians, the marginal density $p(\mathbf{X})$ is then a continuous mixture of Gaussians, which is a rather expressive model.

1.5.2 Gradient of the decoder

As with most deep learning models, the VAE is usually trained using backpropagation and some form of stochastic gradient descent. For this purpose, the gradients of the ELBO with respect to both the decoder parameters $\boldsymbol{\theta}_d$ and encoder parameters $\boldsymbol{\theta}_e$ must be computable using automatic differentiation. As we see next, computing the former poses no significant difficulties.

Let us assume that $p(\mathbf{X} \mid \mathbf{z}; \boldsymbol{\theta}_d)$ is as defined in equation (1.25) since this is the version that will be considered later in this document. Under this setting,

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_d} \mathrm{ELBO}(\boldsymbol{\theta}_e, \boldsymbol{\theta}_d) &= \nabla_{\boldsymbol{\theta}_d} \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} \mid \boldsymbol{X}_i; \boldsymbol{\theta}_e)} \left[\log \frac{p(\boldsymbol{X}_i \mid \mathbf{z}; \boldsymbol{\theta}_d) p(\mathbf{z})}{q(\mathbf{z} \mid \boldsymbol{X}_i; \boldsymbol{\theta}_e)} \right] \\
&= \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} \mid \boldsymbol{X}_i; \boldsymbol{\theta}_e)} \left[\nabla_{\boldsymbol{\theta}_d} \log \frac{p(\boldsymbol{X}_i \mid \mathbf{z}; \boldsymbol{\theta}_d) p(\mathbf{z})}{q(\mathbf{z} \mid \boldsymbol{X}_i; \boldsymbol{\theta}_e)} \right] \\
&= \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} \mid \boldsymbol{X}_i; \boldsymbol{\theta}_e)} \left[\nabla_{\boldsymbol{\theta}_d} \log p(\boldsymbol{X}_i \mid \mathbf{z}; \boldsymbol{\theta}_d) \right] \\
&= -\frac{1}{2\sigma_d^2} \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} \mid \boldsymbol{X}_i; \boldsymbol{\theta}_e)} \left[\nabla_{\boldsymbol{\theta}_d} ||\boldsymbol{X}_i - \boldsymbol{\mu}_d(\mathbf{z}; \boldsymbol{\theta}_d)||^2 \right],
\end{aligned} \tag{1.26}$$

where the second equality is due to the linearity of expectations and the last one comes by plugging in the formula of the Gaussian density and discarding constant terms. This expression makes it clear that the decoder is trained to reconstruct the original image given

its latent code by minimizing the L^2 distance between the original image and its reconstruction, just like a traditional (i.e. non-variational) autoencoder. Finally, the expectation in equation (1.26) is replaced by its empirical approximation with one sample:

$$\nabla_{\boldsymbol{\theta}_d}\mathrm{ELBO}(\boldsymbol{\theta}_e,\boldsymbol{\theta}_d) \approx -\frac{1}{2\sigma_d^2}\sum_{i=1}^{n}\nabla_{\boldsymbol{\theta}_d}||\boldsymbol{X}_i-\boldsymbol{\mu}_d(\boldsymbol{z}_i;\boldsymbol{\theta}_d)||^2, \tag{1.27}$$

where the $\boldsymbol{z}_i$ are sampled from $q(\mathbf{z}\mid\boldsymbol{X}_i;\boldsymbol{\theta}_e)$, which is Gaussian, so this process is computationally easy and efficient.

1.5.3 Gradient of the encoder

Finding the gradient of the ELBO with respect to the encoder parameters is not as easy since the distribution we are taking the expectation with respect to depends on the encoder parameters. Thus, the expectation and the gradient operator do not commute and we get:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}_e}\mathrm{ELBO}(\boldsymbol{\theta}_e,\boldsymbol{\theta}_d) &= \nabla_{\boldsymbol{\theta}_e}\sum_{i=1}^{n}\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\boldsymbol{X}_i;\boldsymbol{\theta}_e)}\left[\log\frac{p(\boldsymbol{X}_i\mid\mathbf{z};\boldsymbol{\theta}_d)p(\mathbf{z})}{q(\mathbf{z}\mid\boldsymbol{X}_i;\boldsymbol{\theta}_e)}\right]\\ &= \nabla_{\boldsymbol{\theta}_e}\sum_{i=1}^{n}\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\boldsymbol{X}_i;\boldsymbol{\theta}_e)}\left[\log p(\boldsymbol{X}_i\mid\mathbf{z};\boldsymbol{\theta}_d)\right]-\nabla_{\boldsymbol{\theta}_e}\sum_{i=1}^{n}D_{\mathrm{KL}}(q(\mathbf{z}\mid\boldsymbol{X}_i;\boldsymbol{\theta}_e)\|p(\mathbf{z})),\end{aligned} \tag{1.28}$$

where the KL divergence is between two Gaussian distributions and hence can be computed analytically as:

$$D_{\mathrm{KL}}(q(\mathbf{z}\mid\boldsymbol{X}_i;\boldsymbol{\theta}_e)\|p(\mathbf{z})) = \sum_{j=1}^{l}\left(\mu_j^{(i)^2}+\sigma_j^{(i)^2}-1-2\log\sigma_j^{(i)}\right), \tag{1.29}$$

where l is the dimension of the latent vector $\mathbf{z}$ and $\mu_j^{(i)}$ and $\sigma_j^{(i)}$ denote the j-th elements of the vectors $\boldsymbol{\mu}_e(\boldsymbol{X}_i;\boldsymbol{\theta}_e)$ and $\boldsymbol{\sigma}_e(\boldsymbol{X}_i;\boldsymbol{\theta}_e)$, respectively.

Now, all that remains is the computation of the gradient with respect to the first term in equation (1.28). Note that:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}_e}\sum_{i=1}^{n}\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\boldsymbol{X}_i;\boldsymbol{\theta}_e)}[\log p(\boldsymbol{X}_i\mid\mathbf{z};\boldsymbol{\theta}_d)] &= \nabla_{\boldsymbol{\theta}_e}\sum_{i=1}^{n}\int q(\mathbf{z}\mid\boldsymbol{X}_i;\boldsymbol{\theta}_e)\log p(\boldsymbol{X}_i\mid\mathbf{z};\boldsymbol{\theta}_d)\,\mathrm{d}\mathbf{z}\\ &= \sum_{i=1}^{n}\int\nabla_{\boldsymbol{\theta}_e}q(\mathbf{z}\mid\boldsymbol{X}_i;\boldsymbol{\theta}_e)\log p(\boldsymbol{X}_i\mid\mathbf{z};\boldsymbol{\theta}_d)\,\mathrm{d}\mathbf{z}\\ &= \sum_{i=1}^{n}\int q(\mathbf{z}\mid\boldsymbol{X}_i;\boldsymbol{\theta}_e)\nabla_{\boldsymbol{\theta}_e}\log q(\mathbf{z}\mid\boldsymbol{X}_i;\boldsymbol{\theta}_e)\log p(\boldsymbol{X}_i\mid\mathbf{z};\boldsymbol{\theta}_d)\,\mathrm{d}\mathbf{z}\end{aligned}$$

$$= \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\boldsymbol{X}_i;\boldsymbol{\theta}_e)} \left[\nabla_{\boldsymbol{\theta}_e} \log q(\mathbf{z} \mid \boldsymbol{X}_i;\boldsymbol{\theta}_e) \log p(\boldsymbol{X}_i \mid \mathbf{z};\boldsymbol{\theta}_d)\right], \tag{1.30}$$

where the third equality comes by applying the log derivative trick ($\nabla u = u\nabla \log(u)$). In the last expression, the gradient operator appears inside the expectation and so, in principle, it can be approximated using an empirical expectation:

$$\nabla_{\boldsymbol{\theta}_e} \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\boldsymbol{X}_i;\boldsymbol{\theta}_e)}[\log p(\boldsymbol{X}_i \mid \mathbf{z};\boldsymbol{\theta}_d)] \approx \frac{1}{m} \sum_{i=1}^{n} \sum_{j=1}^{m} \nabla_{\boldsymbol{\theta}_e} \log q(\boldsymbol{z}_{i,j} \mid \boldsymbol{X}_i;\boldsymbol{\theta}_e) \log p(\boldsymbol{X}_i \mid \boldsymbol{z}_{i,j};\boldsymbol{\theta}_d), \tag{1.31}$$

where the $\boldsymbol{z}_{i,j}$ are sampled from $q(\mathbf{z} \mid \boldsymbol{X}_i;\boldsymbol{\theta}_e)$. The problem with this approach is that, early in training, $p(\boldsymbol{X}_i \mid \boldsymbol{z}_{i,j};\boldsymbol{\theta}_d) \approx 0$ and hence $\log p(\boldsymbol{X}_i \mid \boldsymbol{z}_{i,j};\boldsymbol{\theta}_d)$ is negative and has a very large magnitude. This, combined with the fact that $\log q(\boldsymbol{z}_{i,j} \mid \boldsymbol{X}_i;\boldsymbol{\theta}_e)$ can be either positive or negative, leads to a very large variance in the gradient estimator. Reducing the variance implies using a large number of samples m which in turn leads to an increased computational cost.

The *reparameterization trick* is an efficient alternative that overcomes this problem. Note that, if $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; 0, \boldsymbol{I})$, then $\mathbf{z} \triangleq \boldsymbol{\sigma}_e \odot \boldsymbol{\epsilon} + \boldsymbol{\mu}_e \sim \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_e, \mathrm{diag}(\boldsymbol{\sigma}_e)^2)$, where $\odot$ denotes the element-wise product. Thus,

$$\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{X};\boldsymbol{\theta}_e)}[\log p(\mathbf{X} \mid \mathbf{z};\boldsymbol{\theta}_d)] = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{\epsilon};0,\boldsymbol{I})} \left[\log p(\mathbf{X} \mid \boldsymbol{\sigma}_e \odot \boldsymbol{\epsilon} + \boldsymbol{\mu}_e;\boldsymbol{\theta}_d)\right] \tag{1.32}$$

and therefore

$$\nabla_{\boldsymbol{\theta}_e} \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\boldsymbol{X}_i;\boldsymbol{\theta}_e)}[\log p(\boldsymbol{X}_i \mid \mathbf{z};\boldsymbol{\theta}_d)] = \sum_{i=1}^{n} \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{\epsilon};0,\boldsymbol{I})} \left[\nabla_{\boldsymbol{\theta}_e} \log p(\boldsymbol{X}_i \mid \boldsymbol{\sigma}_e^{(i)} \odot \boldsymbol{\epsilon} + \boldsymbol{\mu}_e^{(i)};\boldsymbol{\theta}_d)\right], \tag{1.33}$$

where $\boldsymbol{\mu}_e^{(i)}$ and $\boldsymbol{\sigma}_e^{(i)}$ are shorthand notations for $\boldsymbol{\mu}_e(\boldsymbol{X}_i;\boldsymbol{\theta}_e)$ and $\boldsymbol{\sigma}_e(\boldsymbol{X}_i;\boldsymbol{\theta}_e)$, respectively. Note that in this case there is no product of logarithms inside the expectation and so the high-variance problem is mitigated. The most common approach is again to use an empirical approximation with one sample, which yields:

$$\nabla_{\boldsymbol{\theta}_e} \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\boldsymbol{X}_i;\boldsymbol{\theta}_e)}[\log p(\boldsymbol{X}_i \mid \mathbf{z};\boldsymbol{\theta}_d)] \approx \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}_e} \log p(\boldsymbol{X}_i \mid \boldsymbol{\sigma}_e^{(i)} \odot \boldsymbol{\epsilon}_i + \boldsymbol{\mu}_e^{(i)};\boldsymbol{\theta}_d), \tag{1.34}$$

where $\boldsymbol{\epsilon}_i$ is sampled from a standard Gaussian.

1.5.4 Loss function

Most deep learning libraries follow the principle that a model is trained to minimize a given loss function. In the case of the VAE, we have derived the (approximate) gradients directly. Finding an expression for the corresponding loss (i.e. for a loss function that yields the same gradients) is straightforward, but we write it for completeness and also because it provides useful insights into the behavior of the VAE.

From equations (1.27), (1.28), and (1.34), it should be clear that maximizing the ELBO is equivalent to the minimization of:

$$L_{\text{VAE}}(\boldsymbol{\theta}_e, \boldsymbol{\theta}_d) \triangleq \sum_{i=1}^{n} ||\boldsymbol{X}_i - \boldsymbol{\mu}_d(\boldsymbol{z}_i; \boldsymbol{\theta}_d)||^2 + \lambda \sum_{i=1}^{n} D_{\text{KL}}(q(\mathbf{z} \mid \boldsymbol{X}_i; \boldsymbol{\theta}_e) \| p(\mathbf{z})), \tag{1.35}$$

where $\boldsymbol{z}_i \triangleq \boldsymbol{\sigma}_e^{(i)} \odot \boldsymbol{\epsilon}_i + \boldsymbol{\mu}_e^{(i)}$, $\boldsymbol{\epsilon}_i$ is sampled from a standard Gaussian, $\lambda = 2\sigma_d^2 > 0$ is a hyperparameter, and $D_{\text{KL}}(q(\mathbf{z} \mid \boldsymbol{X}_i; \boldsymbol{\theta}_e) \| p(\mathbf{z}))$ is as written in equation (1.29). Equation (1.35) shows that the VAE is trained to satisfy to two objectives: i) reconstructing the original samples and ii) approximating the distribution of latent codes to the prior $p(\mathrm{z})$, which is a standard Gaussian. If both are accomplished, new images $\boldsymbol{X}$ can be generated efficiently by sampling a latent code $\boldsymbol{z}$ from a standard Gaussian and performing a forward pass through de decoder, a procedure that corresponds to forward sampling in the chain $\mathrm{z} \rightarrow \mathbf{X}$.

Chapter 2

Networked data streams

Some parts of this chapter were originally published in or adapted from:

> [10] A. Allahdadi, D. Pernes, J. S. Cardoso, and R. Morla, "Hidden Markov models on a self-organizing map for anomaly detection in 802.11 wireless networks," *Neural Computing and Applications*, pp. 1–18, 2021 (presented in Section 2.2)
>
> [11] D. Pernes and J. S. Cardoso, "SpaMHMM: Sparse mixture of hidden Markov models for graph connected entities," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–10 (presented in Section 2.3)

In [10], Diogo Pernes worked primarily on the mathematical derivation of the algorithm and provided some advice about the experimental protocol. Allahdadi was responsible for motivating the application, researching related work, conducting the experimental evaluation, and generating the wireless simulation dataset. Cardoso and Morla supervised the work.

2.1 Introduction

A broad range of real-life settings can be well modeled by an arbitrary number of network-connected entities that share and interact in the same medium and generate data streams in real-time. The streams produced by each of these entities form a set of time series with both intra and inter-correlations between them. In neuroimaging studies, the brain can be regarded as a network: a connected system where nodes, or units, represent different specialized regions and links, or connections, represent communication pathways. From a functional perspective, communication is coded by temporal dependence between the activities of different brain areas (De Vico Fallani et al. [12]). Also team sports intrinsically

involve fast, complex and interdependent events among a set of entities (the players), which interact as a team (Tora et al. [13], Theagarajan et al. [14]). The emergence of vehicular networks is also generating an ever-increasing amount of network data (Cheng et al. [15]), where interactions between neighboring vehicles may be exploited to build more accurate and reliable learning algorithms. Thus, in all these scenarios the behavior of each individual entity is better understood if its context information (i.e. the behavior of the neighboring instances) is leveraged. However, the extraction of knowledge from these streams to support the decision-making process is still challenging. Moreover, conventional algorithms that assume ability to store and centralize all the data in memory at the same time are impractical for many applications (Gama and Gaber [16]).

Modeling the generative process of distributed stream data is an unsupervised learning problem and, hence, a model can be learned directly from the large amounts of data that might be continuously produced or gathered at each network connected entity, without the requirement of any special human supervision or annotation. Moreover, generative models are powerful tools for a wide variety of problems that arise naturally in networked data streams, like anomaly and novelty detection, sequence forecasting, clustering, and network simulation. Hence, generative models for stream data have been developed and applied in several previous works (e.g. Laxman et al. [17], Hayat and Hashemi [18], Hofmann and Sick [19]). However, to the best of our knowledge, this problem is seldom explored in the distributed setting.

Given the distributed nature of network data and the high information rate that those streams could have, we argue that such generative model and/or the associated learning algorithm should ideally satisfy the following properties:

1. Learning and inferring distributedly, i.e. each entity should be able to update its own model and perform inference on it without observing the streams or the models associated with the remaining entities.

2. Learning online, i.e. in real-time and without the requirement of storing the whole dataset in memory.

3. Leveraging contextual information by incorporating prior knowledge about similarities and dissimilarities among sets of entities.

4. The model should be applicable to a wide variety of distributed stream data, of diverse nature.

In this chapter, we present two solutions to this problem, each with its own advantages and disadvantages. Both are inspired by the concept of sparse representations, which expresses a signal/model f, defined over an independent variable x, as a linear combination of a few atoms from a prespecified and overcomplete dictionary of size m:

$$f(x) = \sum_{z=1}^{m} s_z \phi_z(x), \tag{2.1}$$

where $\phi_z(x)$ are the atoms and only a few of the scalars s_z are non-zero, providing a sparse representation of $f(x)$. Distributed sparse representation (Baron et al. [20]) is an extension of the standard version that considers networks with k nodes. At each node, the signal sensed at the same node has its sparsity property because of its intra-correlation, while, for networks with multiple nodes, signals received at different nodes also exhibit strong intercorrelation. The intra and inter-correlations lead to a joint sparse model. An interesting scenario in distributed sparse representation, which we exploit here, is when all signals/models share the common support but with different non-zero coefficients.

Our contributions in this chapter are summarized as follows: i) we extend an existing model based on a combination of self-organizing maps (SOM) and HMMs and evaluate it in the context of anomaly detection in Wi-Fi networks (Section 2.2); ii) we present a novel algorithm that learns an entity-dependent model based on a mixture of shared HMMs (Section 2.3); iii) we discuss how the two models intersect each other and, importantly, how the latter can be viewed as a particular case of a general family of generative models for distributed data (Section 2.4).

2.2 Hidden Markov Models on a Self-Organizing Map for Anomaly Detection in 802.11 Wireless Networks

2.2.1 Introduction

In large-scale 802.11 wireless networks, acquiring a baseline knowledge of the entire infrastructure is not straightforward. Owing to the time-varying and physically distributed nature of these networks, learning the usage characteristics of access points (APs), users, and locations becomes more and more challenging. The wireless channel conditions evolve over time, as does the usage behavior of the wireless users in different parts of the network. Thus, wireless users are likely to suffer from many types of connectivity and performance problems, e.g. interference, intermittent connections, or authentication

failures. To constantly ensure that wireless users have reliable connections, network managers require tools and techniques to monitor the network, identify the problems, and resolve them efficiently.

Naive approaches, e.g. using a single HMM common to all APs, lose the flexibility to adapt to the specificities of each AP, while using one HMM per AP, trained independently from the others, fails to leverage the relations between observations of neighboring APs. An HMM initialized with a universal background model (HMM-UBM, Allahdadi et al. [21]) is an improvement in the right direction; however, the relations between APs are only used in the initial phase, when one trains the UBM to then initialize the individual HMM models for each AP. Thereafter, the individual HMMs evolve independently, only benefiting from the AP's own data. Although this is a very sensible approach in the biometrics and speech modeling fields, where HMM-UBM has been used to train robust user-specific models, in our setting, it fails to properly explore the dependencies between data from APs with similar behavior.

In the current work, we focus on the actual proximity of APs as a determining factor in connectivity and performance problems. Anomalous cases, e.g. across-AP-vicinity interference, AP overloads, or AP shutdown/halt, could eventually affect the usage behavior of the other APs in the neighborhood. To consider such behavioral changes in the local area and their respective influences, we employ the synergistic approach of the self-organizing hidden Markov model map (SOHMMM) to exploit the semantic connectivity between adjacent HMMs.

The self-organizing map is an artificial neural network that defines a nonlinear transformation from the input space to the set of nodes in the output space (Somervuo [22]). Each node or neuron in the SOM is associated with a model of the input space. Through an unsupervised-learning process, the models are tuned and organized in a lattice topology according to the input patterns. In SOHMMM, each neuron is literally associated with an HMM.

The training processes of the SOM and HMM sub-units are, in most cases, disjoint and conducted independently. There are two main approaches regarding these hybrid techniques. The first approach considers the SOM as a front-end processor (e.g. vector quantization, preprocessing, feature extraction); HMMs are then used in the higher processing stages (Somervuo [22], Kurimo and Somervuo [23], Morimoto [24]). The second

approach places the SOM on top of the HMM (Ferles and Stafylopatis [25], Ferles et al. [26], Lebbah et al. [27]).

In SOHMMM, the SOM unsupervised learning approach is well combined with the HMM dynamic programming technique. The structure of both corresponding components is unified in an integrated super-model. The presented online gradient-descent unsupervised learning algorithm is inspired by the SOHMMM algorithm previously proposed by Ferles and Stafylopatis [25] and motivated by Baldi and Chauvin [28]. We extend the model to fit the requirements of our anomaly-detection problem and extend the presented algorithm by Ferles and Stafylopatis [25] to multivariate Gaussian emissions*.

2.2.2 Related work

A number of studies in the literature have integrated the SOM and HMM in different manners. Niina and Dozono [29] proposed the spherical self-organizing map (S-SOM), which uses HMM models as neurons (S-HMM-SOM) to classify the time-series data. This work only considers discrete observations and the model parameters are updated using the Baum-Welch algorithm (Baum [6]). Yamaguchi [30] extended the self-organizing mixture models for multivariate time-series, assuming that the time-series are generated by HMMs. This model, which is called a self-organizing hidden Markov model (SOHMM), uses constrained expectation maximization (EM) for HMM parameter estimation. The self-organization in this work is used for meteorological-state visualization.

In another direction of work, Caridakis et al. [31] present a SOMM-based architecture for hand-gesture recognition. The approach involves a combination of SOMs and Markov models for gesture-trajectory classification. In this work, the neurons on the SOM map correspond to the states of the Markov models. Jaziri et al. [32] also exploit the combination of the SOM and HMM (SOS-HMM: self-organizing structure of HMM) and automatically extract the structure of an HMM without any prior knowledge of the application domain. In this model, the macro-HMM is represented as a graph of macro-states, where each state represents a micro-HMM. In summary, each neuron in the SOM-HMM collaborative architecture is either an HMM by itself or a hidden state. In our work, each particular neuron on the SOM lattice is associated with an HMM.

Lebbah et al. [27] present a probabilistic self-organizing map called PrSOMS for clustering and visualization of dependent and non-identically distributed data. In this model,

*Ferles and Stafylopatis [25] only address the discrete-observation setting.

the SOM learning paradigm to produce topology-preserving maps is combined with the probabilistic-learning scheme of the HMM. The SOM is considered to be a grid, forming a discrete topology in which each cell represents a state; the parameters of the model are estimated by maximizing the likelihood of the sequential data set. Morimoto [24] investigates the effects of meteorological factors on the occurrence of strokes. The authors used the SOM to obtain weather patterns that would serve as states of the HMMs. They showed that HMMs with states given by the SOM are useful for describing a background process of stroke incidence. This approach considers the SOM as a front-end processor.

Ferles and Stafylopatis [25, 33], Ferles et al. [34] apply the fusion and synergy of SOMs and HMMs in biological-molecule studies to meet the increasing requirements imposed by the properties of deoxyribonucleic acid (DNA), ribonucleic acid (RNA), and protein chain molecules. The authors proposed a stochastic unsupervised-learning algorithm based on the integration of the SOM and HMM principles, called self-organizing hidden Markov model map (SOHMMM). The SOHMMM characteristics and capabilities are demonstrated through two series of experiments, based on artificial sequence data and splice-junction gene sequences. However, in these papers, only the discrete-observation setting is addressed. Here, we extend the algorithm for multivariate Gaussian in order to fit the requirements of our anomaly-detection project.

Incremental learning of HMM parameters is the core function of the SOHMMM algorithm, which is based on a stochastic gradient-descent technique. The incremental learning of new data sequences allows the HMM parameters to adapt as new data become available; without having to be retrained from scratch on all the accumulated training data. Various techniques in the literature address this topic. These techniques are classified according to the objective function, optimization technique, and target application, and include the block-wise and symbol-wise learning of parameters. Khreich et al. [35] presented a comprehensive survey of techniques that are suitable for the incremental learning of HMM parameters, mentioning the stochastic gradient-descent technique of the SOHMMM as one of the numerical optimization methods.

Additionally, few efforts exist in the literature that exploit the SOM and HMM for anomaly-detection purposes (Cho [36], Wang et al. [37]). Cho [36] presented an intrusion-detection system in which the SOM determines the optimal measures of audit data and reduces them to an appropriate size for efficient modeling by the HMM. Two types of HMM are applied: a single model for all the users and individual models for each user.

Wang et al. [37] investigated the HMM and the SOM separately as intrusion-detection techniques. The testing results show that the HMM method using the events' transition property outperformed the SOM using the events' frequency property. Regarding the same subject of intrusion detection, the SOM and HMM have a collaborating connection in [36] and competitive roles in [37].

In this work, we intend to benefit from the collaboration of these two techniques (SOM and HMM), as proposed by Ferles and Stafylopatis [33], to extend previous anomaly-detection frameworks applying only HMM (Allahdadi et al. [21, 38], Allahdadi and Morla [39]).

2.2.3 The Self-Organizing Hidden Markov Model Map for Discrete Observations

We start by reviewing SOHMMM as initially formulated by Ferles and Stafylopatis [33]. This model defines a mapping between an observed sequence $\mathbf{x} \triangleq \left(\mathrm{x}^{(1)}, \ldots, \mathrm{x}^{(T)}\right)$ and a two-dimensional lattice of HMMs, which constitute its atoms. Each observation is assumed to be discrete, so $\mathbf{x}^{(t)} \in \{1, \ldots, o\}$, being o the size of the dictionary of observations. The topology of a 2-D lattice of m HMMs is defined by a function $\boldsymbol{r} : \{1, \ldots, m\} \mapsto \mathbb{R}^2$, mapping the indices of the m nodes to the respective coordinates in the plane. Given the lattice topology, a neighborhood function $v : \{1, \ldots, m\}^2 \mapsto [0, \infty)$ can be defined mapping two nodes in the lattice to a scalar representing how close the two atoms are. In SOHMMM, v is a Gaussian kernel:

$$v(z, z') \triangleq \exp\left(-\lambda ||\boldsymbol{r}(z) - \boldsymbol{r}(z')||^2\right), \tag{2.2}$$

where $\lambda > 0$ is a hyperparameter controlling the rate of the decay. Following the idea of SOMs, the SOHMMM algorithm aims to minimize an *energy function*, defined below:

$$E(\boldsymbol{x}; \Theta) \triangleq -\sum_{\mathrm{z}} p(\boldsymbol{x} \mid \mathrm{z}, \Theta) v(\mathrm{z}, z^*), \tag{2.3}$$

where Θ summarizes all parameters of the HMMs, z indexes the m HMMs in the lattice, and $p(\mathbf{x} \mid \mathrm{z})$ is the marginal distribution of observations of a standard HMM:

$$p(\mathbf{x} \mid \mathrm{z}) = \sum_{\mathbf{h}} p(\mathrm{h}^{(1)} \mid \mathrm{z}) p(\mathrm{x}^{(1)} \mid \mathrm{h}^{(1)}, \mathrm{z}) \prod_{t=2}^{T} p(\mathrm{h}^{(t)} \mid \mathrm{h}^{(t-1)}, \mathrm{z}) p(\mathrm{x}^{(t)} \mid \mathrm{h}^{(t)}, \mathrm{z}), \tag{2.4}$$

where $\mathbf{h} = \left(\mathrm{h}^{(1)}, \ldots, \mathrm{h}^{(T)}\right)$ is the sequence of hidden states of the HMM and $\mathrm{h}^{(t)} \in \{1, \ldots, s\}$, being s the number of hidden states. Finally, z^* corresponds to the index of the *winner node* for the observation $\boldsymbol{x}$:

$$z^* \triangleq \underset{z' \in \{1,\ldots,m\}}{\arg\max} \sum_{\mathrm{z}} p(\boldsymbol{x} \mid \mathrm{z}, \Theta) v(\mathrm{z}, z'). \tag{2.5}$$

The set Θ consists of the following parameters:

- the s-dimensional initial state probabilities, $\boldsymbol{\pi}^{(z)}$, where $\pi_h^{(z)} \triangleq p(\mathrm{h}^{(1)} = h \mid \mathrm{z} = z)$, for $h \in \{1, \ldots, s\}$ and $z \in \{1, \ldots, m\}$;
- the $s \times s$ state transition matrices, $\boldsymbol{A}^{(z)}$, where $A_{h,h'}^{(z)} \triangleq p(\mathrm{h}^{(t)} = h' \mid \mathrm{h}^{(t-1)} = h, \mathrm{z} = z)$, for $h, h' \in \{1, \ldots, s\}$ and $z \in \{1, \ldots, m\}$;
- the $s \times o$ emission probability matrices, $\boldsymbol{B}^{(z)}$, where $B_{h,x}^{(z)} \triangleq p(\mathrm{x}^{(t)} = x \mid \mathrm{h}^{(t)} = h, \mathrm{z} = z)$, for $h \in \{1, \ldots, s\}$, $x \in \{1, \ldots, o\}$, and $z \in \{1, \ldots, m\}$.

Thus, $\Theta \triangleq \{(\boldsymbol{\pi}^{(z)}, \boldsymbol{A}^{(z)}, \boldsymbol{B}^{(z)})\}_{z=1}^{m}$, where each triplet $(\boldsymbol{\pi}^{(z)}, \boldsymbol{A}^{(z)}, \boldsymbol{B}^{(z)})$ completely defines one HMM in the lattice. The SOHMMM online learning algorithm corresponds to stochastic gradient descent over the energy function (2.3). Constrained optimization is avoided by reparametrizing the model using softmax functions, so that standard, unconstrained stochastic gradient descent can be performed over the new parameters $\boldsymbol{u}^{(z)}$, $\boldsymbol{W}^{(z)}$, and $\boldsymbol{R}^{(z)}$:

$$\pi_h^{(z)} = \frac{\exp(u_h^{(z)})}{\sum_{h'=1}^{s} \exp(u_{h'}^{(z)})}, \quad A_{h,h'}^{(z)} = \frac{\exp(W_{h,h'}^{(z)})}{\sum_{h''=1}^{s} \exp(W_{h,h''}^{(z)})}, \quad B_{h,x}^{(z)} = \frac{\exp(R_{h,x}^{(z)})}{\sum_{x'=1}^{o} \exp(R_{h,x'}^{(z)})}. \tag{2.6}$$

The full algorithm, comprising the update equations for each parameter, is presented in Algorithm 2.1 and its derivation can be found in Ferles and Stafylopatis [33].

2.2.4 Extending SOHMMM to Gaussian Observations

For the purpose of modeling AP usage data, we should consider a slightly different setting where each observation $\mathbf{x}^{(t)}$ takes values on the d-dimensional plane $\mathbb{R}^d$, rather than being drawn from a discrete and finite set. Hence, now, sequences $\mathbf{X} \triangleq \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)}\right)$ take values on $\mathbb{R}^{d \times T}$. A sensible option, which may be useful in a broad range of applications, is considering the case of multivariate Gaussian emissions, i.e. $\mathbf{x}^{(t)} \mid (\mathrm{h}^{(t)}, \mathrm{z}) \sim \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu}_h^{(z)}, \boldsymbol{\Sigma}_h^{(z)}\right)$, where $\boldsymbol{\mu}_h^{(z)} \in \mathbb{R}^d$ is the mean and $\boldsymbol{\Sigma}_h^{(z)} \in \mathbb{R}^{d \times d}$ is the covariance of the

Algorithm 2.1 SOHMMM learning algorithm for discrete observations [33]

Inputs: the lattice r with m nodes, the hyperparameter λ of the neighborhood function v, the set of initial parameters $\Theta^{(0)}$, the learning rate η, and the number of training iterations `trainIter`.

for $i = 1, \ldots,$ `trainIter` **do**

Observe $\mathbf{x} = \boldsymbol{x}$, with length T.

for $z = 1, \ldots, m$ **do** (forward-backward algorithm)

$\alpha_h^{(z)}(1) \coloneqq \pi_h^{(z)} B_{h,x^{(1)}}^{(z)}$, for $h = 1, \ldots, s$;

$\alpha_h^{(z)}(t) \coloneqq B_{h,x^{(t)}}^{(z)} \sum_{h'=1}^{s} \alpha_{h'}^{(z)}(t-1) A_{h',h}^{(z)}, \quad t = 2, \ldots, T$, for $h = 1, \ldots, s$;

$\beta_h^{(z)}(T) \coloneqq 1$, for $h = 1, \ldots, s$;

$\beta_h^{(z)}(t) \coloneqq B_{h,x^{(t+1)}}^{(z)} \sum_{h'=1}^{s} \beta_{h'}^{(z)}(t+1) A_{h,h'}^{(z)}$, for $t = T-1, \ldots, 1$ and $h = 1, \ldots, s$.

end for

$z^* \coloneqq \arg\max_{z' \in \{1,\ldots,m\}} \sum_{z=1}^{m} p(\boldsymbol{x} \mid z, \Theta^{(i-1)}) v(z, z')$.

for $z = 1, \ldots, m$ **do**

$u_h^{(z)} \coloneqq u_h^{(z)} + \eta \pi_h^{(z)} \left[B_{h,x^{(1)}}^{(z)} \beta_1^{(z)}(h) - p(\boldsymbol{x} \mid z, \Theta^{(i-1)}) \right]$, for $h = 1, \ldots, s$;

$W_{h,h'}^{(z)} \coloneqq W_{h,h'}^{(z)} + \eta v(z, z^*) A_{h',h}^{(z)} \sum_{t=1}^{T-1} \left[\alpha_h^{(z)}(t) B_{h,x^{(t+1)}}^{(z)} \beta_{t+1}^{(z)}(h') - \alpha_h^{(z)}(t) \beta_h^{(z)}(t) \right]$, for $h, h' = 1, \ldots, s$;

$R_{h,x'}^{(z)} \coloneqq R_{h,x'}^{(z)} + \eta v(z, z^*) \sum_{t=1}^{T} \left[\mathbf{1}_{x^{(t)}=x'} \alpha_h^{(z)}(t) \beta_h(t) - B_{h,x'}^{(z)} \alpha_h^{(z)}(t) \beta_h^{(z)}(t) \right]$, for $h = 1, \ldots, s$ and $x' = 1, \ldots, o$;

$\pi_h^{(z)} \coloneqq \exp(u_h^{(z)}) \Big/ \sum_{h'=1}^{s} \exp(u_{h'}^{(z)})$, for $h = 1, \ldots, s$;

$A_{h,h'}^{(z)} \coloneqq \exp(W_{h,h'}^{(z)}) \Big/ \sum_{h''=1}^{s} \exp(W_{h,h''}^{(z)})$, for $h, h' = 1, \ldots, s$;

$B_{h,x'}^{(z)} \coloneqq \exp(R_{h,x'}^{(z)}) \Big/ \sum_{x''=1}^{o} \exp(R_{h,x''}^{(z)})$, for $h = 1, \ldots, s$ and $x' = 1, \ldots, o$.

end for

$\Theta^{(i)} \coloneqq \left\{ (\boldsymbol{u}^{(z)}, \boldsymbol{W}^{(z)}, \boldsymbol{R}^{(z)}) \right\}_{z=1}^{m}$.

end for

Gaussian component corresponding to state h in the lattice node z. These means and covariances replace the emission probability matrices $\boldsymbol{B}^{(z)}$ defined for the case of discrete observations.

We shall now derive the necessary readjustments in Algorithm 2.1. Let us denote by $\boldsymbol{x}$ the observation at a given time t (i.e. $\mathbf{x}^{(t)} = \boldsymbol{x}$) and consider the corresponding state $\mathrm{h}^{(t)}$ and the node z as fixed at h and z, respectively. Thus, we use the notation $p(\boldsymbol{x} \mid h, z)$ as a short for $p(\mathbf{x}^{(t)} = \boldsymbol{x} \mid \mathrm{h}^{(t)} = h, \mathrm{z} = z)$. We have:

$$\begin{aligned}
\frac{\partial p(\boldsymbol{X} \mid z)}{\partial p(\boldsymbol{x} \mid h, z)} &= \frac{\partial}{\partial p(\boldsymbol{x} \mid h, z)} \left[\sum_{\mathbf{h}} p(\boldsymbol{X}, \mathbf{h} \mid z) \right] \\
&= \frac{\partial}{\partial p(\boldsymbol{x} \mid h, z)} \left[\sum_{\mathbf{h}} p(\mathrm{h}^{(1)} \mid \mathrm{z}) \prod_{t'=1}^{T} p(\mathrm{h}^{(t')} \mid \mathrm{h}^{(t'-1)}, \mathrm{z}) p(\mathbf{x}^{(t')} \mid \mathrm{h}^{(t')}, \mathrm{z}) \right] \\
&= \frac{1}{p(\boldsymbol{x} \mid h, z)} \sum_{\mathbf{h}: \mathrm{h}^{(t)} = h} p(\boldsymbol{X}, \mathbf{h} \mid z) \\
&= \frac{p(\boldsymbol{X}, h \mid z)}{p(\boldsymbol{x} \mid h, z)} \\
&= \frac{\alpha_t^{(z)}(h) \beta_t^{(z)}(h)}{p(\boldsymbol{x} \mid h, z)}.
\end{aligned} \tag{2.7}$$

Note that, for discrete observations, $p(x \mid h, z) = B_{h,x}^{(z)}$. However, when we consider the multivariate Gaussian case,

$$\begin{aligned}
p(\boldsymbol{x} \mid h, z) &= \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu}_h^{(z)}, \boldsymbol{\Sigma}_h^{(z)}\right) \\
&= \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma}_h^{(z)})}} \exp\left(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu}_h^{(z)})^\top \boldsymbol{\Sigma}_h^{(z)^{-1}} (\boldsymbol{x} - \boldsymbol{\mu}_h^{(z)}) \right),
\end{aligned} \tag{2.8}$$

where the covariance matrix $\boldsymbol{\Sigma}_h^{(z)}$ must be positive definite. This constraint can be easily guaranteed by reparametrizing this matrix as $\boldsymbol{\Sigma}_h^{(z)} = \boldsymbol{S}_h^{(z)} \boldsymbol{S}_h^{(z)^\top}$, where $\boldsymbol{S}_h^{(z)} \in \mathbb{R}^{d \times d}$ is constraint-free (although assumed to be non-singular). The required gradients are:

$$\nabla_{\boldsymbol{\mu}_h^{(z)}} p(\boldsymbol{x} \mid h, z) = p(\boldsymbol{x} \mid h, z) (\boldsymbol{S}\boldsymbol{S}^\top)^{-1} (\boldsymbol{x} - \boldsymbol{\mu}), \tag{2.9}$$

$$\nabla_{\boldsymbol{S}_h^{(z)}} p(\boldsymbol{x} \mid h, z) = p(\boldsymbol{x} \mid h, z) \boldsymbol{S}^{-1} \left[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^\top (\boldsymbol{S}\boldsymbol{S}^\top)^{-1} - I_d \right], \tag{2.10}$$

where we have abbreviated $\boldsymbol{\mu}_h^{(z)}$ as $\boldsymbol{\mu}$ and $\boldsymbol{S}_h^{(z)}$ as $\boldsymbol{S}$ to alleviate the notation. Now, by the chain rule,

$$\begin{aligned}
\nabla_{\boldsymbol{\mu}_h^{(z)}} p(\boldsymbol{X} \mid z) &= \sum_{t=1}^{T} \frac{\partial p(\boldsymbol{X} \mid z)}{\partial p(\boldsymbol{x}^{(t)} \mid h, z)} \nabla_{\boldsymbol{\mu}_h^{(z)}} p(\boldsymbol{x}^{(t)} \mid h, z) \\
&= \sum_{t=1}^{T} \frac{\alpha_t^{(z)}(h)\beta_t^{(z)}(h)}{p(\boldsymbol{x}^{(t)} \mid h, z)} \nabla_{\boldsymbol{\mu}_h^{(z)}} p(\boldsymbol{x}^{(t)} \mid h, z) \\
&= \sum_{t=1}^{T} \alpha_t^{(z)}(h)\beta_t^{(z)}(h)(\boldsymbol{S}\boldsymbol{S}^\top)^{-1}(\boldsymbol{x}^{(t)} - \boldsymbol{\mu}), \qquad (2.11) \\
\nabla_{\boldsymbol{S}_h^{(z)}} p(\boldsymbol{X} \mid z) &= \sum_{t=1}^{T} \alpha_t^{(z)}(h)\beta_t^{(z)}(h)\boldsymbol{S}^{-1}\left[(\boldsymbol{x}^{(t)} - \boldsymbol{\mu})(\boldsymbol{x}^{(t)} - \boldsymbol{\mu})^\top (\boldsymbol{S}\boldsymbol{S}^\top)^{-1} - I_d\right]. \qquad (2.12)
\end{aligned}$$

Finally, by applying the chain rule once again, we get the desired gradients of the energy function E with respect to $\boldsymbol{\mu}_h^{(z)}$ and $\boldsymbol{S}_h^{(z)}$:

$$\begin{aligned}
\nabla_{\boldsymbol{\mu}_h^{(z)}} E(\boldsymbol{X}) &= \frac{\partial E}{\partial p(\boldsymbol{X} \mid z)} \nabla_{\boldsymbol{\mu}_h^{(z)}} p(\boldsymbol{X} \mid z) \\
&= -v(z, z^*) \nabla_{\boldsymbol{\mu}_h^{(z)}} p(\boldsymbol{X} \mid z) \\
&= -v(z, z^*) \sum_{t=1}^{T} \alpha_t^{(z)}(h)\beta_t^{(z)}(h)(\boldsymbol{S}\boldsymbol{S}^\top)^{-1}(\boldsymbol{x}^{(t)} - \boldsymbol{\mu}), \qquad (2.13) \\
\nabla_{\boldsymbol{S}_h^{(z)}} E(\boldsymbol{X}) &= -v(z, z^*) \sum_{t=1}^{T} \alpha_t^{(z)}(h)\beta_t^{(z)}(h)\boldsymbol{S}^{-1}\left[(\boldsymbol{x}^{(t)} - \boldsymbol{\mu})(\boldsymbol{x}^{(t)} - \boldsymbol{\mu})^\top (\boldsymbol{S}\boldsymbol{S}^\top)^{-1} - I_d\right]. \qquad (2.14)
\end{aligned}$$

Given equations (2.13) and (2.14), it is straightforward to adapt Algorithm 2.1 to our setting. For completeness, the full algorithm is provided in Algorithm 2.2.

2.2.5 Experiments

In this section, we consider two types of experiments to analyze the capabilities of the SOHMMM algorithm for nonlinear projection and unsupervised clustering. We validate first the accuracy and convergence of the SOHMMM using *synthetic data*, and then explore its significance and efficiency in anomaly detection using *wireless simulation data*.

2.2.5.1 Synthetic Data

In this experiment, we generate observations from two reference HMMs, with one-third of the observations coming from one of the models, and the remaining two-thirds from the other reference model. Then, we train a SOHMMM with six nodes, randomly initialized, with the data from the reference models. It is expected that the SOHMMM nodes would

Algorithm 2.2 SOHMMM learning algorithm for Gaussian observations

1: **Inputs:** Same as in Algorithm 2.1.

2: **for** $i = 1, \ldots,$ `trainIter` **do**

3: Observe $\mathbf{X} = \boldsymbol{X}$, with length T.

4: Proceed as in lines 4–8 of Algorithm 2.1, replacing all occurrences of $B^{(z)}_{h,x^{(t)}}$ with $\mathcal{N}\left(\boldsymbol{x}^{(t)}; \boldsymbol{\mu}_h^{(z)}, \boldsymbol{S}_h^{(z)} \boldsymbol{S}_h^{(z)\top}\right)$.

5: $z^* := \arg\max_{z' \in \{1,\ldots,m\}} \sum_{z=1}^{m} p(\boldsymbol{X} \mid z, \Theta^{(i-1)}) v(z, z')$.

6: **for** $z = 1, \ldots, m$ **do**

7: Proceed as in lines 12 and 13 of Algorithm 2.1, replacing all occurrences of $B^{(z)}_{h,x^{(t)}}$ with $\mathcal{N}\left(\boldsymbol{x}^{(t)}; \boldsymbol{\mu}_h^{(z)}, \boldsymbol{S}_h^{(z)} \boldsymbol{S}_h^{(z)\top}\right)$ and all occurences of $\boldsymbol{x}$ with $\boldsymbol{X}$;

8: $\boldsymbol{\mu}_h^{(z)} := \boldsymbol{\mu}_h^{(z)} + \eta v(z, z^*) \sum_{t=1}^{T} \alpha_t^{(z)}(h) \beta_t^{(z)}(h) (\boldsymbol{S}_h^{(z)} \boldsymbol{S}_h^{(z)\top})^{-1} (\boldsymbol{x}^{(t)} - \boldsymbol{\mu}_h^{(z)})$, for $h = 1, \ldots, s$;

9: $$\boldsymbol{S}_h^{(z)} := \boldsymbol{S}_h^{(z)} + \eta v(z, z^*) \sum_{t=1}^{T} \alpha_t^{(z)}(h) \beta_t^{(z)}(h) \boldsymbol{S}_h^{(z)^{-1}} \cdot \cdot \left[(\boldsymbol{x}^{(t)} - \boldsymbol{\mu}_h^{(z)}) (\boldsymbol{x}^{(t)} - \boldsymbol{\mu}_h^{(z)})^\top (\boldsymbol{S}_h^{(z)} \boldsymbol{S}_h^{(z)\top})^{-1} - I_d \right], \text{ for } h = 1, \ldots, s;$$

10: Proceed as in lines 15 and 16 of Algorithm 2.1.

11: **end for**

12: $\Theta^{(i)} := \left\{ (\boldsymbol{u}^{(z)}, \boldsymbol{W}^{(z)}, (\boldsymbol{\mu}_h^{(z)}, \boldsymbol{S}_h^{(z)})_{h=1}^{s}) \right\}_{z=1}^{m}$.

13: **end for**

converge to the reference models, with the majority of nodes grouping around the dominant reference model. The SOHMMM nodes (HMMs initialized randomly) are organized in a 2×3 rectangular lattice, as displayed in Figure 2.1. This figure shows the positions and connections of the HMMs. The Euclidean distance between adjacent HMMs is set to 1 and the other distances are computed accordingly. For example, the distance between *hmm0* and *hmm4* is 2, and between *hmm0* and *hmm3* is $\sqrt{2}$.

In order to perform this experiment, we need a quantitative measurement of similarity between two HMMs, so that we can evaluate how close each HMM in the lattice is from each of the two reference HMMs. It is known that two HMMs representing the same marginal distribution of observations may be parametrized by different parameters

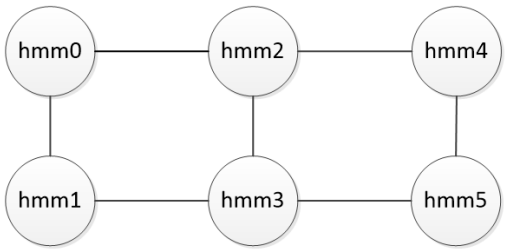


FIGURE 2.1: 2×3 rectangular lattice.

(Rabiner [40]), so the dissimilarity should not be computed in parameter space. An alternative metric to compare two HMMs z and z', proposed by Juang and Rabiner [41], is the following divergence function:

$$D(z, z') = \frac{1}{t} \left[\log p(\boldsymbol{X} \mid z) - \log p(\boldsymbol{X} \mid z') \right], \tag{2.15}$$

where $\boldsymbol{X}$ has length t and is sampled from $p(\mathbf{X} \mid z)$. Equation (2.15) is a Monte Carlo approximation of the Kullback-Leibler divergence between two HMMs and therefore measures how well model z' matches observations generated by model z, relative to how well model z matches observations generated by itself. Since this relationship is asymmetric, we use the following symmetrized version instead:

$$D_s(z, z') = \frac{D(z, z') + D(z', z)}{2}. \tag{2.16}$$

Figure 2.2 demonstrates the initial and final distances between the random HMMs and the reference HMMs, upon applying the SOHMMM algorithm. The training set contains 60 observation sequences from the two reference HMMs (the *ref0* model generates 40 observation sequences and *ref1* generates 20). We trained the random HMMs with these observation sequences. Our main goal in analyzing the aforementioned distance is to examine how the observation sequences influence the random HMMs and whether they maintain their proximity while moving in different directions. Measuring this distance will indicate whether the random HMMs converge to a specific reference HMM.

As the heatmap plots of Figure 2.2 show, *hmm4* is initially assigned to the *ref1* cluster, as there is a shorter distance between these two models. After applying the algorithm, *hmm4* becomes closer to *ref0*. As another example, *hmm3* is closer to *ref0*, before and after applying the algorithm; however, the overall distance to the reference models decreases and *hmm3* moves closer to both of them while maintaining its relative distance to the

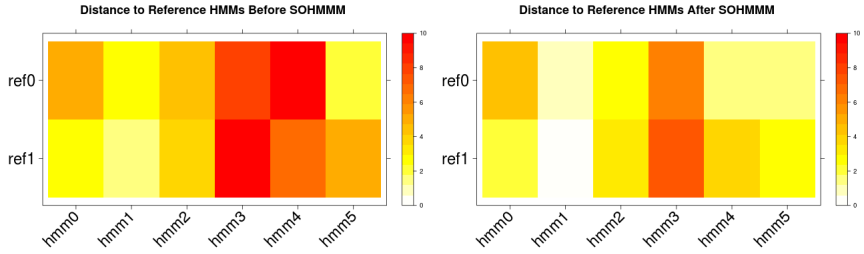


FIGURE 2.2: Monte Carlo approximation of the Kullback-Leibler divergence between the random and reference HMMs before and after applying the SOHMMM algorithm.

references.

The minimum distance between a given random HMM and the reference HMMs defines the cluster that the HMM belongs to. In this experiment, the random HMMs belong to the $(ref1, ref1, ref1, ref0, ref1, ref0)$ and $(ref1, ref1, ref0, ref0, ref0, ref0)$ clusters, before and after applying the algorithm, respectively. The latter clusters show that $\{hmm2, hmm3, hmm4, hmm5\}$ belong to the $ref0$ cluster, the dominant reference models; $\{hmm0, hmm1\}$ are assigned to the $ref1$ cluster, the reference model generating less data. Having analyzed the Euclidean distances of the HMM nodes, Figure 2.1 shows that nearby HMMs are grouped in the same cluster.

We repeated the experiment for various sequence lengths and different types of neighborhood, in terms of the vicinity. We selected a large neighborhood and updated the winner neuron, in addition to all the other neurons in the SOM lattice, based on their relative distances in the lattice. The large-neighborhood experiment employs the SOHMMM algorithm in its original form, so all neurons are updated. In the medium-neighborhood experiment, only the winner HMM and the adjacent neighbors are updated. In the small-neighborhood experiment, only the winner HMM is updated (this setting is known as Z-SOHMMM).

Figure 2.3 displays the Monte Carlo approximation of the Kullback-Leibler divergence between the random and reference HMMs, with large, medium, and small neighborhood sizes, and sequence lengths of 10, 20, and 50. In each experiment, the sum of the distance between the random HMMs and the assigned reference HMM is computed. As Figure 2.3 shows, the sum of the distances to the assigned reference HMMs decreases as the sequence length increases. In addition, the lower distance values are obtained in the large-neighborhood experiment rather than the medium- and the small-neighborhood experiments. This shows that the SOHMMM algorithm provides a better estimation of the

reference models by including all the HMMs in the vicinity. The heatmap plots of Figure 2.2 belong to the large-neighborhood experiment with a sequence length of 50, which produced the best overall distance estimate according to Figure 2.3.

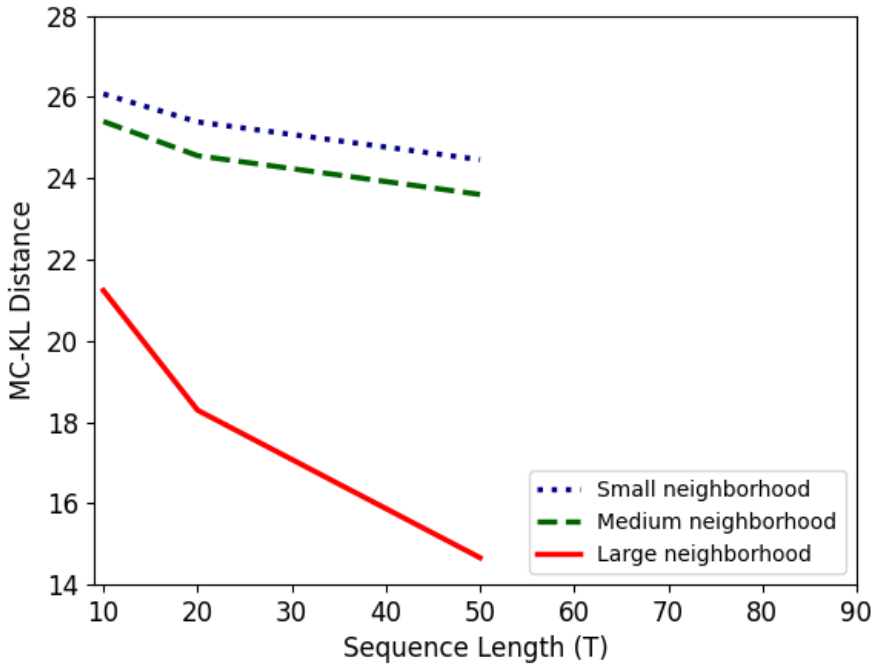


FIGURE 2.3: Monte Carlo approximation of the Kullback-Leibler divergence between the random and reference HMMs with different neighborhood sizes and sequence lengths.

2.2.5.2 Wireless Simulation Data

We implemented a set of wireless simulations using the OMNeT++ [42] simulator together with the INET framework [43]. OMNeT++ is a C++-based discrete-event simulator (DES) for modeling communication networks, multiprocessors, and other distributed or parallel systems. As in any discrete-event simulator, events in OMNeT++ take place at discrete instances in time, and they take zero time to occur. It is assumed that nothing important occurs between two consecutive events. Thus, the simulation time depends on the order of events in the events queue; it could take more or less time than the real CPU time, based on the number of nodes, amount of traffic transferred, and other network details. In our experiment, with the current number of nodes (10 access points (APs) and 100 wireless stations (STAs)) and the defined traffic plan, 10 minutes of simulation time took around two hours of CPU time.

The normal scenario contains 10 APs and 100 STAs. Each STA is initially associated with one of the available APs, depending on its location. During the simulation, the STAs are handed over to other APs, based on their mobility models, when moving around the simulation ground. Furthermore, according to the defined traffic plans, each node sends and receives packets to the existing servers. Figure 2.4 shows images of a normal scenario, the location of the APs, STAs, and servers, and the location of the wireless stations. More details on the mobility models of the wireless stations, traffic generation, available servers, and path-loss models can be found in [21].

Each sequence contains 40 consecutive time-slots of 15s simulation time each. Our simulation consists of one normal scenario and four anomalous scenarios: AP shutdown/ halt, AP overload, noise, and flash crowd. We simulated 15 instances of 3000s of simulation time for the normal scenario, and five instances of 3000s of simulation time for each of the anomalous scenarios. In the following paragraphs, we explain how we employed these data for training and testing the SOHMMM algorithm. Data is divided into training and test sets according to a 80%/20% random split.

In the SOHMMM adapted to the problem of anomaly detection in AP usage data, the self-organization learning process is elaborated as follows. The models associated with the SOM neurons are three-state HMMs (according to the best practice found by Allahdadi et al. [21], Allahdadi and Morla [39]). As the new sequence arrives, the HMM with the lowest energy is selected as the winner model, as specified by equation (2.5). In the AP usage data, as the newly arrived sequence belongs to a pre-determined AP, in most cases, the winner model is related to the same AP that originated the observation sequence. However, this is not always the case, and the competition defines the winner HMM eventually. Thereafter, the HMM model of the winner AP z^* is updated by the new data sequence, and the HMMs in the neighborhood of the winner AP are also updated through a gradient descent step over the energy function (2.3).

At this point, it should be noted that the APs in the vicinity of the winner AP are updated, to some extent, relative to their proximity (or similarity) to the winner AP. In our case, the neighborhood area has an irregular shape and contains the first-level adjacent APs in the vicinity of the winner AP. As the locations of the APs are already determined in the wireless ground (Figure 2.4), the Euclidean distances between all APs are calculated and kept in a complete graph. Then, a filter is applied to update only APs with a certain distance from the winner AP (in this case, half of the maximum distance between AP pairs

in the distance graph).

During the learning process, the nearby HMMs, up to a certain distance, activate each other to gain some information from the new observation sequence. As the distant HMMs should gain only an insignificant amount of information from the new observation sequence, we utilized the aforementioned filter to avoid updating very distant HMMs, and quicken the process. The neighborhood function that we used in this experiment is a slightly modified version of equation (2.2) (with $\lambda = 10$), based on the relative distances of the winner AP and all its adjacent neighbors:

$$v(z, z') \triangleq \exp\left(-10 \frac{||\boldsymbol{r}(z) - \boldsymbol{r}(z')||^2}{\sum_{z'' \in \mathrm{Adj}(z')} ||\boldsymbol{r}(z'') - \boldsymbol{r}(z')||^2}\right), \tag{2.17}$$

where $\mathrm{Adj}(z')$ denotes the set of nodes adjacent to node z'.

We compared the SOHMMM algorithm to two other approaches: i) hidden Markov model initialized with universal background model (HMM-UBM), addressed in detail in [21], and ii) the SOHMMM algorithm with a zero neighborhood, which utilizes the incremental-learning part of the SOHMMM algorithm, without updating the HMMs in the vicinity (Z-SOHMMM).

Equivalent amounts of normal and anomalous data were used to initialize the HMM-UBM model: Five weeks of normal data and one week for each of the five anomalous cases, i.e. ten weeks of data overall. Each week contains five working days. In the anomalous cases, the time and period of the anomalies are different for each day. We used one more week of each scenario to train the model and then used the one remaining week of each case to test the model. The results are represented as receiver operating characteristic (ROC) curves on the test set.

The following anomalous scenarios were simulated:

- AP shutdown/halt, where the AP simply stops working (e.g. due to power failure);
- AP overload, which happens when excessive channel utilization occurs as a consequence of excessive traffic by a number of wireless users;
- noise in the wireless communication channel;
- flash crowd, where multiple users associate (flash crowd arrival) or dis-associate from the same AP almost simultaneously;
- miscellaneous anomalies, where multiple anomalies occur on the same day.

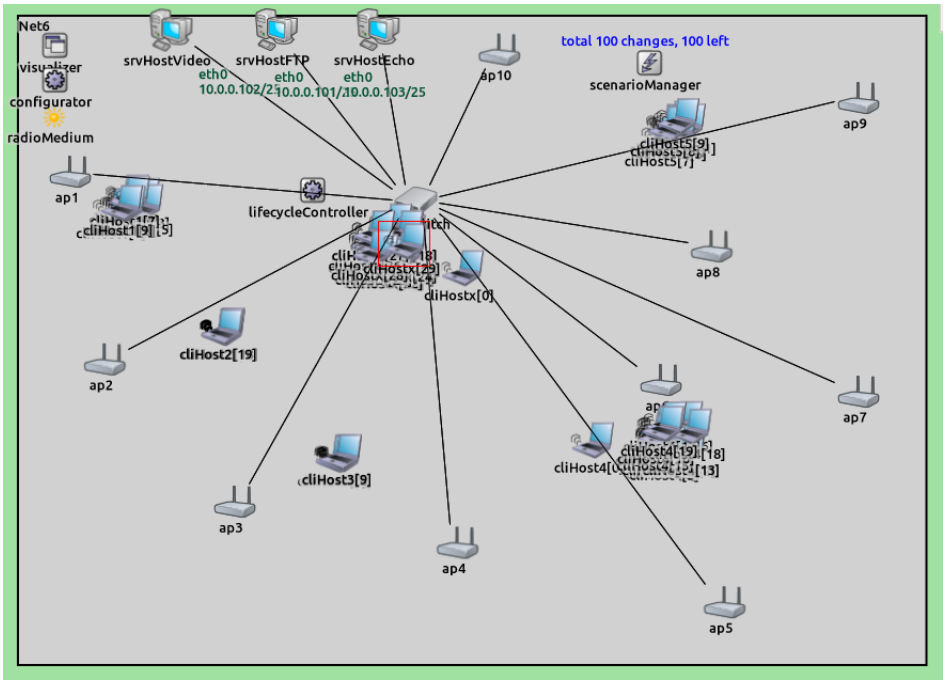


FIGURE 2.4: Wireless network simulated in OMNeT++/INET.

Table 2.1 presents the results for anomaly detection in the aforementioned scenarios. In this table, the AUC values of each anomalous scenario are presented for the HMM-UBM, Z-SOHMMM, and SOHMMM models. The anomalous APs are also indicated. The higher AUC values show that SOHMMM outperforms the other models since it is better at discriminating normal and anomalous samples.

Please refer to Allahdadi et al. [10] for more details about the experiments and further experimental results.

		HU	ZS	S
AP Shutdown/Halt	AP2	0.91	0.95	**0.99**
	AP4	0.71	0.93	**1.00**
AP Overload	AP2	0.99	0.99	**1.00**
Noise	AP2	0.78	0.62	**0.82**
Flash Crowd - Arrival	AP2	0.80	0.91	**0.94**
Flash Crowd - Departure	AP2	0.96	**0.97**	**0.97**
Miscellaneous Anomalies	AP2	0.74	0.73	**0.88**
	AP3	0.69	0.69	**0.71**

TABLE 2.1: AUC values for each model in each anomalous scenario. The best results for each scenario are in bold. (HU– HMM-UBM, ZS– Z-SOHMMM, S– SOHMMM)

2.2.6 Conclusion

In the current work, we applied a hybrid integration of the self-organizing map (SOM) and the hidden Markov model (HMM), called the SOHMMM, for anomaly detection in 802.11 wireless networks. We further extended the online gradient-descent unsupervised-learning algorithm of the SOHMMM for multivariate Gaussian emissions. We employed this algorithm specifically for anomaly detection in 802.11 wireless AP usage data.

The experimental analysis investigated two main data sets: *synthetic data* and *wireless simulation data*. In the synthetic data analysis, we generated six random HMMs and trained them with the observation sequences of two reference HMMs with predefined parameters. We then estimated the distance between HMMs as the Monte Carlo approximation of the Kullback-Leibler divergence, and computed the final HMM clusters, based on the minimum distance between the random HMMs and reference HMMs. We repeated the experiment for various sequence lengths and different neighborhood sizes and showed that, for the large neighborhood, the SOHMMM algorithm provided a better estimation of the reference models, including all the HMMs in the vicinity. Moreover, we presented the decay of the learning rate and the convergence of the loss function.

In the analysis regarding the wireless simulation data, we showed how the SOHMMM algorithm improved the anomaly-detection accuracy and sensitivity, compared to the HMM-UBM and Z-SOHMMM techniques in the AP shutdown/halt, AP overload, noise, and flash-crowd anomalous scenarios. We further investigated the combination of several anomalies in one observation sequence as miscellaneous anomalies and showed that the SOHMMM was capable of detecting contrasting anomalous cases while the HMM-UBM was not.

2.3 SpaMHMM: Sparse Mixture of Hidden Markov Models for Graph Connected Entities

2.3.1 Overview

Inspired by the formulation of equation (2.1), we propose to model the generative distribution of the data coming from each of the k nodes of a network as a sparse mixture obtained from a dictionary of generative distributions. Specifically, we shall model the distribution for each node as a sparse mixture over a 'large' shared dictionary of HMMs,

where each HMM corresponds to an individual atom from the dictionary. The field knowledge about the similarities between nodes is summarized in an affinity matrix. The objective function of the learning process promotes reusing HMM atoms between similar nodes. We now formalize these ideas.

2.3.2 Model formulation

2.3.2.1 Definition

Assume we have a set of nodes $\mathcal{Y} = \{1, \ldots, k\}$ connected by an undirected weighted graph $\mathcal{G}$, expressed by a symmetric matrix $\boldsymbol{G} \in \mathbb{R}^{k \times k}$. These nodes thus form a network, in which the weights are assumed to represent degrees of affinity between each pair of nodes (i.e. the greater the edge weight, the more the respective nodes *like* to agree). The nodes y in the graph produce d-dimensional sequences $\mathbf{X} = \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)}\right)$, $\mathbf{x}^{(t)} \in \mathbb{R}^d$, whose conditional distribution we shall model using a mixture of HMMs:

$$p(\mathbf{X} \mid \mathrm{y}) = \sum_{\mathrm{z}} p(\mathrm{z} \mid \mathrm{y}) p(\mathbf{X} \mid \mathrm{z}), \tag{2.18}$$

where $\mathrm{z} \in \{1, \ldots, m\}$ is a latent random variable, being m the size of the mixture. This is a particular realization of equation (2.1) where f is the probability density function $p(\mathbf{X} \mid \mathrm{y})$ and the coefficients s_z correspond to the probabilities $p(\mathrm{z} = z \mid \mathrm{y})$. Here, $p(\mathbf{X} \mid \mathrm{z})$ is the marginal distribution of observations of a standard first-order homogeneous HMM:

$$p(\mathbf{X} \mid \mathrm{z}) = \sum_{\mathbf{h}} p(\mathrm{h}^{(1)} \mid \mathrm{z}) p(\mathbf{x}^{(1)} \mid \mathrm{h}^{(1)}, \mathrm{z}) \prod_{t=2}^{T} p(\mathrm{h}^{(t)} \mid \mathrm{h}^{(t-1)}, \mathrm{z}) p(\mathbf{x}^{(t)} \mid \mathrm{h}^{(t)}, \mathrm{z}), \tag{2.19}$$

where $\mathbf{h} = \left(\mathrm{h}^{(1)}, \ldots, \mathrm{h}^{(T)}\right)$ is the sequence of hidden states of the HMM and $\mathrm{h}^{(t)} \in \{1, \ldots, s\}$, being s the number of hidden states. Note that the factorization in equation (2.18) enforces conditional independence between the sequence $\mathbf{X}$ and the node y, given the latent variable z. This is a key assumption of this model since in this way the distributions for the observations in the nodes in $\mathcal{Y}$ share the same dictionary of HMMs, promoting parameter sharing among the k mixtures. The Bayesian network representing this model is presented in Figure 2.5.

2.3.2.2 Inference

Given an observed sequence $\boldsymbol{X}$ and its corresponding node $y \in \mathcal{Y}$, the inference problem here consists in finding the likelihood $p(\mathbf{X} = \boldsymbol{X} \mid \mathrm{y} = y)$ (from now on, abbreviated as

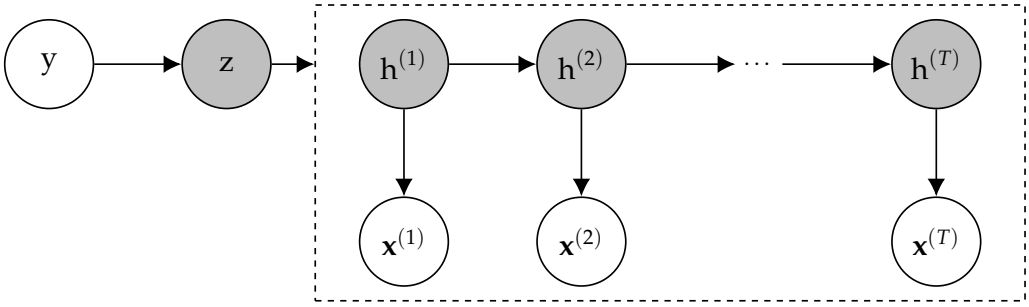


FIGURE 2.5: Representation of the model as a Bayesian network. The node z is a parent of all nodes inside the dashed box (the connections were omitted for clarity). Gray nodes are used for latent variables.

$p(\boldsymbol{X} \mid y)$) as defined by equations (2.18) and (2.19). The marginals $p(\boldsymbol{X} \mid \mathrm{z})$ of each HMM in the mixture can be computed efficiently, in $O(Ts^2)$ time, using Algorithm 1.1. Then, $p(\boldsymbol{X} \mid y)$ is obtained by applying equation (2.18), so inference in the overall model is done in at most $O(Ts^2m)$ time. As we shall see, however, the mixtures we obtain after learning will often be sparse (see Section 2.3.2.3), leading to even smaller time complexity.

2.3.2.3 Learning

Given an i.i.d. dataset consisting of n tuples $(\boldsymbol{X}_i, y_i)$ of sequences of observations $\boldsymbol{X}_i = \left(\boldsymbol{x}_i^{(1)}, \ldots, \boldsymbol{x}_i^{(T_i)}\right)$ and their respective nodes $y_i \in \mathcal{Y}$, the model defined by equations (2.18) and (2.19) may be easily trained using the Expectation-Maximization (EM) algorithm (Dempster et al. [44]), (locally) maximizing the usual log-likelihood objective:

$$J(\theta) \triangleq \sum_{i=1}^{n} \log p(\boldsymbol{X}_i \mid y_i; \Theta), \tag{2.20}$$

where Θ represents all model parameters, namely:

1. the m-dimensional mixture coefficients, $\boldsymbol{\alpha}^{(y)}$, where $\alpha_z^{(y)} \triangleq p(\mathrm{z} = z \mid \mathrm{y} = y)$, for $z \in \{1, \ldots, m\}$ and $y \in \{1, \ldots, k\}$;

2. the s-dimensional initial state probabilities, $\boldsymbol{\pi}^{(z)}$, where $\pi_h^{(z)} \triangleq p(\mathrm{h}^{(1)} = h \mid \mathrm{z} = z)$, for $h \in \{1, \ldots, s\}$ and $z \in \{1, \ldots, m\}$;

3. the $s \times s$ state transition matrices, $\boldsymbol{A}^{(z)}$, where $A_{h,h'}^{(z)} \triangleq p(\mathrm{h}^{(t)} = h' \mid \mathrm{h}^{(t-1)} = h, \mathrm{z} = z)$, for $h, h' \in \{1, \ldots, s\}$ and $z \in \{1, \ldots, m\}$;

4. the emission probability means, $\boldsymbol{\mu}_h^{(z)} \in \mathbb{R}^d$, for $z \in \{1, \ldots, m\}$ and $h \in \{1, \ldots, s\}$;

5. the emission probability diagonal covariance matrices, $\Sigma_h^{(z)} = \mathrm{diag}(\sigma_h^{(z)})^2$, where $\boldsymbol{\sigma}_h^{(z)} \in \mathbb{R}^d$, for $z \in \{1, \ldots, m\}$ and $h \in \{1, \ldots, s\}$.

Here, we are assuming that the emission probabilities $p(\mathrm{x}^{(t)} \mid \mathrm{h}^{(t)}, \mathrm{z})$ are Gaussian with diagonal covariances. This introduces almost no loss of generality since the extension of this work to discrete observations or other types of continuous emission distributions is straightforward.

The procedure to maximize objective (2.20) using EM is described in Algorithm 2.3. The update formulas follow from the standard EM procedure and can be obtained by viewing this model as a Bayesian network or by following the derivation detailed in Section A.1.1. However, the objective (2.20) does not take advantage of the known structure of $\mathcal{G}$. In order to exploit this information, we introduce a regularization term, maximizing the following objective instead:

$$\begin{aligned} J_r(\Theta) &\triangleq \frac{1}{n}\sum_{i=1}^{n} \log p(\boldsymbol{X}_i \mid y_i; \Theta) + \frac{\lambda}{2} \sum_{\substack{y,y'=1,\\ y' \neq y}}^{k} G_{y,y'} \mathbb{E}_{\mathrm{z}\sim p(\mathrm{z}|y;\Theta)}[p(\mathrm{z} \mid y'; \Theta)] \\ &= \frac{1}{n}\sum_{i=1}^{n} \log p(\boldsymbol{X}_i \mid y_i; \Theta) + \frac{\lambda}{2} \sum_{\substack{y,y'=1,\\ y' \neq y}}^{k} G_{y,y'} \boldsymbol{\alpha}^{(y)\top} \boldsymbol{\alpha}^{(y')}, \end{aligned} \tag{2.21}$$

where $\lambda \geq 0$ controls the relative weight of the two terms in the objective. Note that this regularization term favors nodes connected by edges with large positive weights to have similar mixture coefficients and thus share mixture components. On the other hand, nodes connected by edges with large negative weights will tend to have orthogonal mixture coefficients, being described by disjoint sets of components. These observations agree with our prior assumption that the edge weights express degrees of similarity between each pair of nodes. Proposition 2.1 formalizes these statements and enlightens interesting properties about the expectations $\mathbb{E}_{\mathrm{z}\sim p(\mathrm{z}|y)}[p(\mathrm{z} \mid y')]$.

Proposition 2.1. *For any integer $m > 1$, let $\mathbb{P}_m$ be the set of all probability distributions over the set $\{1, \ldots, m\}$. We have:*

1. $\min_{p,q\in\mathbb{P}_m} \mathbb{E}_{\mathrm{z}\sim p}[q(\mathrm{z})] = 0$;
2. $\arg\min_{p,q\in\mathbb{P}_m} \mathbb{E}_{\mathrm{z}\sim p}[q(\mathrm{z})] = \{p, q \in \mathbb{P}_M \mid \forall z \in \{1, \ldots, m\} : p(\mathrm{z} = z)q(\mathrm{z} = z) = 0\}$;
3. $\max_{p,q\in\mathbb{P}_m} \mathbb{E}_{\mathrm{z}\sim p}[q(\mathrm{z})] = 1$;
4. $\arg\max_{p,q\in\mathbb{P}_m} \mathbb{E}_{\mathrm{z}\sim p}[q(\mathrm{z})] = \{p, q \in \mathbb{P}_M \mid \exists z \in \{1, \ldots, m\} : p(\mathrm{z} = z) = q(\mathrm{z} = z) = 1\}$.

Proof. By the definition of expectation, for any $p, q \in \mathbb{P}_m$,

$$\mathbb{E}_{z\sim p}[q(z)] = \sum_{z} p(z)q(z). \tag{2.22}$$

Statements 1 and 2 follow immediately from the fact that every term on the right-hand side of (2.22) is non-negative and $m > 1$. For the remaining, we rewrite (2.22) as the dot product of two m-dimensional vectors $\boldsymbol{\alpha}_p$ and $\boldsymbol{\alpha}_q$, representing the two distributions p and q, respectively, and we use the following linear algebra inequalities to build an upper bound for this expectation:

$$\mathbb{E}_{z\sim p}[q(z)] = \boldsymbol{\alpha}_p^\top \boldsymbol{\alpha}_q \leq ||\boldsymbol{\alpha}_p||_2 ||\boldsymbol{\alpha}_q||_2 \leq ||\boldsymbol{\alpha}_p||_1 ||\boldsymbol{\alpha}_q||_1 = 1, \tag{2.23}$$

where $||\cdot||_1$ and $||\cdot||_2$ are the L^1 and L^2 norms, respectively. The equality $\mathbb{E}_{z\sim p}[q(z)] = 1$ holds if p and q are chosen from the set defined in statement 4, where the distributions p and q are the same and they are non-zero for a single assignment of z. This proves statement 3. Now, to prove statement 4, it suffices to show that there are no other maximizers. The first inequality in (2.23) is transformed into equality if and only if $\boldsymbol{\alpha}_p = \boldsymbol{\alpha}_q$, which means $p \equiv q$. The second inequality becomes equality when the L^1 and L^2 norms of the vectors coincide, which happens if and only if the vectors have only one non-zero component, concluding the proof. □

Specifically, given two distinct nodes $y, y' \in \mathcal{Y}$, if $G_{y,y'} > 0$, the regularization term for these nodes is maximum (and equal to $G_{y,y'}$) when the mixtures for these two nodes are the same and have one single active component (i.e. one mixture component whose coefficient is non-zero). On the contrary, if $G_{y,y'} < 0$, the term is maximized (and equal to zero) when the mixtures for the two nodes do not share any active components. In both cases, though, we conclude from Proposition 2.1 that we are favoring sparse mixtures. We see sparsity as an important feature since it allows the size m of the dictionary of models to be large and therefore expressive without compromising our rationale that the observations in a given node are well modeled by a mixture of only a few HMMs. This way, some components will specialize in describing the behavior of some nodes, while others will specialize on different nodes. Moreover, sparse mixtures yield faster inference, more interpretable models and (possibly) less overfitting. By setting $\lambda = 0$, we clearly get the initial objective (2.20), where inter-node correlations are modeled only via parameter sharing. As $\lambda \to \infty$, two interesting scenarios may be anticipated. If $G_{y,y'} > 0$, $\forall y, y' \in \mathcal{Y}$, all nodes will tend to share the same single mixture component, i.e. we would be learning

one single HMM to describe the whole network. If $G_{y,y'} < 0$, $\forall y, y' \in \mathcal{Y}$, and $m \geq K$, each node would tend to learn its own HMM model independently from all the others. Again, in both scenarios, the obtained mixtures are sparse.

The objective function (2.21) can still be maximized via EM (see details in Section A.1.2). However, the introduction of the regularization term in the objective makes it impossible to find a closed form solution for the update formula of the mixture coefficients. Thus, in the M-step, we need to resort to gradient ascent to update these parameters. In order to ensure that the gradient ascent iterative steps lead to admissible solutions, we adopt the following reparametrization from Yang et al. [45]:

$$\alpha_z^{(y)} = \frac{\max(0, \beta_z^{(y)})^2}{\sum_{z'=1}^{m} \max(0, \beta_{z'}^{(y)})^2}, \tag{2.24}$$

and so we can treat $\boldsymbol{\beta}^{(y)}$ as an unconstrained parameter vector. This reparametrization clearly resembles the softmax function, but, contrary to that one, admits sparse outputs. The squared terms in equation (2.24) aim only to make the optimization more stable. The optimization steps for the objective (2.21) using this reparametrization are described in Algorithm 2.4.

2.3.3 Experiments

The model was developed on top of the library hmmlearn (Lebedev [46]) for Python, which implements inference and unsupervised learning for the standard HMM using a wide variety of emission distributions. Both learning and inference use the hmmlearn API, with the appropriate adjustments for our models. For reproducibility purposes, we make our source code, pre-trained models and the datasets publicly available*.

We evaluate four different models in our experiments: a model consisting of a single HMM (denoted as 1-HMM) trained on sequences from all graph nodes; a model consisting of k HMMs trained independently (denoted as k-HMM), one for each graph node; a mixture of HMMs (denoted as MHMM) as defined in this work (equations (2.18) and (2.19)), trained to maximize the usual log-likelihood objective (2.20); a mixture of HMMs (denoted as SpaMHMM) as the previous one, trained to maximize our regularized objective (2.21).

Models 1-HMM, k-HMM and MHMM will be our baselines. We shall compare the performance of these models with that of SpaMHMM and, for the case of MHMM, we

*https://github.com/dpernes/spamhmm

Algorithm 2.3 EM algorithm for the mixture without regularization (MHMM).

1: **Inputs:** The training set, consisting of n tuples $(\boldsymbol{X}_i, y_i)$, a set of initial parameters $\Theta^{(0)}$, and the number of training iterations `trainIter`.

2: **for** $j = 1, \ldots,$ `trainIter` **do**

3: **E-step:**

4: $n_y \coloneqq \sum_{i=1}^{n} \mathbf{1}_{y_i=y}$, for $y = 1, \ldots, k$;

5: Obtain the mixture posteriors $\eta_i^{(z)} \coloneqq p(z \mid \boldsymbol{X}_i, y_i)$, for $i = 1, \ldots, n$ and $z = 1, \ldots, m$, by computing $\tilde{\eta}_i^{(z)} \coloneqq p(\boldsymbol{X}_i \mid z)p(z \mid y_i)$ and normalizing it;

6: Obtain the state posteriors $\gamma_{i,h}^{(z)}(t) \coloneqq p(\mathrm{h}^{(t)} = h \mid z, \boldsymbol{X}_i)$ and $\xi_{i,h,h'}^{(z)}(t) \coloneqq p(\mathrm{h}^{(t-1)} = h, \mathrm{h}^{(t)} = h' \mid z, \boldsymbol{X}_i)$, for $i = 1, \ldots, n$, $t = 1, \ldots, T_i$, $z = 1, \ldots, m$, and $h, h' = 1, \ldots, s$, as done in Algorithm 1.1.

7: **M-step:**

8: $\alpha_z^{(y)} \coloneqq \frac{\sum_{i=1}^{n} \eta_i^{(z)} \mathbf{1}_{y_i=y}}{n_y}$, for $y = 1, \ldots, k$ and $z = 1, \ldots, m$, obtaining $\boldsymbol{\alpha}_y$;

9: $\pi_h^{(z)} \coloneqq \frac{\sum_{i=1}^{n} \eta_i^{(z)} \gamma_{i,h}^{(z)}(0)}{\sum_{i=1}^{n} \eta_i^{(z)}}$, for $z = 1, \ldots, m$ and $h = 1, \ldots, s$, obtaining $\boldsymbol{\pi}^{(z)}$;

10: $A_{h,h'}^{(z)} \coloneqq \frac{\sum_{i=1}^{n} \eta_i^{(z)} \sum_{t=1}^{T_i} \xi_{i,h,h'}^{(z)}(t)}{\sum_{i=1}^{n} \eta_i^{(z)} \sum_{t=0}^{T_i-1} \gamma_{i,h}^{(z)}(t)}$, for $z = 1, \ldots, m$, and $h, h' = 1, \ldots, s$, obtaining $\boldsymbol{A}^{(z)}$;

11: $\boldsymbol{\mu}_h^{(z)} \coloneqq \frac{\sum_{i=1}^{n} \eta_i^{(z)} \sum_{t=1}^{T_i} \gamma_{i,h}^{(z)}(t) \boldsymbol{x}_i^{(t)}}{\sum_{i=1}^{n} \eta_i^{(z)} \sum_{t=1}^{T_i} \gamma_{i,h}^{(z)}(t)}$, for $z = 1, \ldots, m$ and $h = 1, \ldots, s$;

12: $\boldsymbol{\sigma}_h^{(z)^2} \coloneqq \frac{\sum_{i=1}^{n} \eta_i^{(z)} \sum_{t=1}^{T_i} \gamma_{i,h}^{(z)}(t) (\boldsymbol{x}_i^{(t)} - \boldsymbol{\mu}_h^{(z)})^2}{\sum_{i=1}^{n} \eta_i^{(z)} \sum_{t=1}^{T_i} \gamma_{i,h}^{(z)}(t)}$, for $z = 1, \ldots, m$ and $h = 1, \ldots, s$;

13: $\Theta^{(j)} \coloneqq \bigcup_{y,z,h} \left\{ \boldsymbol{\alpha}^{(y)}, \boldsymbol{\pi}^{(z)}, \boldsymbol{A}^{(z)}, \boldsymbol{\mu}_h^{(z)}, \boldsymbol{\sigma}_h^{(z)} \right\}$.

14: **end for**

shall also verify if SpaMHMM actually produces sparser mixtures in general, as argued in Section 2.3.2.3. In order to ensure a fair comparison, we train models with approximately the same number of possible state transitions. Hence, given an MHMM or SpaMHMM with m mixture components and s states per component, we train a 1-HMM with $\approx s\sqrt{m}$ states and a k-HMM with $\approx s\sqrt{m/k}$ states per HMM. We initialize the mixture coefficients in MHMM and SpaMHMM randomly, while the state transition matrices and the initial state probabilities are initialized uniformly. Means are initialized using k-means, with k equal to the number of hidden states in the HMM, and covariances are initialized with the diagonal of the training data covariance. Models 1-HMM and k-HMM are trained using

Algorithm 2.4 EM algorithm for the mixture with regularization (SpaMHMM).

1: **Inputs:** The training set, consisting of n tuples $(\boldsymbol{X}_i, y_i)$, the matrix $\boldsymbol{G}$ describing the graph $\mathcal{G}$, the regularization hyperparameter λ, a set of initial parameters $\Theta^{(0)}$, the number of training iterations `trainIter`, the number of gradient ascent iterations `mIter` to perform on each M-step, and the learning rate ρ to perform gradient ascent over the mixture coefficients.

2: **for** $j = 1, \ldots,$ `trainIter` **do**

3: **E-step:** same as in Algorithm 2.3.

4: **M-step:**

5: **for** $l = 1, \ldots,$ `mIter` **do**

6: $\psi_z^{(y)} := \frac{1}{n} \sum_{i=1}^{n} \left(\eta_i^{(z)} - \alpha_z^{(y)} \right) \mathbf{1}_{y_i = y}$, for $y = 1, \ldots, k$ and $z = 1, \ldots, m$;

7: $\omega_z^{(y)} := \alpha_z^{(y)} \sum_{y'=1}^{k} G_{y',y} \left(\alpha_z^{(y')} - \boldsymbol{\alpha}^{(y')\top} \boldsymbol{\alpha}^{(y)} \right) \mathbf{1}_{y' \neq y}$, for $y = 1, \ldots, k$ and $z = 1, \ldots, m$;

8: $\delta_z^{(y)} := \frac{2}{\beta_z^{(y)}} \left(\psi_z^{(y)} + \lambda \omega_z^{(y)} \right) \mathbf{1}_{\beta_z^{(y)} > 0}$, for $y = 1, \ldots, k$ and $m = 1, \ldots, z$;

9: $\beta_z^{(y)} := \beta_z^{(y)} + \rho \delta_z^{(y)}$, for $y = 1, \ldots, k$ and $z = 1, \ldots, m$, obtaining $\boldsymbol{\beta}^{(y)}$;

10: $\alpha_z^{(y)} := \frac{\max(0, \beta_z^{(y)})^2}{\sum_{z'=1}^{m} \max(0, \beta_{z'}^{(y)})^2}$, for $y = 1, \ldots, k$ and $z = 1, \ldots, m$, , obtaining $\boldsymbol{\alpha}^{(y)}$.

11: **end for**

12: Proceed as in lines 9–12 of Algorithm 2.3;

13: $\Theta^{(j)} := \bigcup_{h,y,z} \left\{ \boldsymbol{\beta}^{(y)}, \boldsymbol{\pi}^{(z)}, \boldsymbol{A}^{(z)}, \boldsymbol{\mu}_h^{(z)}, \sigma_h^{(z)} \right\}$.

14: **end for**

the Baum-Welch algorithm, MHMM is trained using Algorithm 2.3 and SpaMHMM is trained using Algorithm 2.4. However, we opted to use Adam (Kingma and Ba [47]) instead of *vanilla* gradient ascent in the inner loop of Algorithm 2.4, since its per-parameter learning rate proved to be beneficial for faster convergence.

2.3.4 Anomaly detection in Wi-Fi networks

A typical Wi-Fi network infrastructure is constituted by k access points (APs) distributed in a given space. Network users can seamlessly switch between these APs, typically connecting to the closest one. There is a wide variety of anomalies that can occur during the operation of such network and their automatic detection is, therefore, of great importance

for future mitigation plans. Some anomalous behaviors are: overloaded APs, failed or crashed APs, persistent radio frequency interference between adjacent APs, authentication failures, etc. However, obtaining reliable ground truth annotation of these anomalies in entire wireless networks is costly and time-consuming. Under these circumstances, using data obtained through realistic network simulations is a common practice.

In order to evaluate our model in the aforementioned scenario, we have followed the procedure of Allahdadi et al. [21], performing extensive network simulations in a typical Wi-Fi network setup (IEEE 802.11 WLANg 2.4 GHz in infrastructure mode) using OMNeT++ [42] and INET [43] simulators. Our network consists of 10 APs and 100 users accessing it. The pairwise distances between APs are known and fixed. Each sequence contains information about the traffic in a given AP during 10 consecutive hours and is divided in time slots of 15 minutes without overlap. Thus, every sequence has the same length, equal to 40 samples (time slots). Each sample contains the following 7 features: the number of unique users connected to the AP, the number of sessions within the AP, the total duration (in seconds) of association time of all current users, the number of octets transmitted and received in the AP and the number of packets transmitted and received in the AP. Anomalies typically occur for a limited amount of time within the whole sequence. However, in this experiment, we label a sequence as "anomalous" if there is at least one anomaly period in the sequence and we label it as "normal" otherwise. One of the simulations includes normal data only, while the remaining include both normal and anomalous sequences. In order to avoid contamination of normal data with anomalies that may occur simultaneously in other APs, we used the data of the normal simulation for training (150 sequences) and the remaining data for testing (378 normal and 42 anomalous sequences).

In a Wi-Fi network, when users move in the covered area, they disconnect from one AP and immediately connect to another in the vicinity. As such, the traffic in adjacent APs may be expected to be similar. Following this idea, the weight $G_{y,y'}$, associated with the edge connecting nodes y and y' in graph $\mathcal{G}$, was set to the inverse of the distance between APs y and y' and normalized so that $\max_{y,y'} G_{y,y'} = 1$. Following Allahdadi et al. [21], sequences were preprocessed by subtracting the mean and dividing by the standard deviation and applying PCA, reducing the number of features to 3. For MHMM, we did 3-fold cross-validation of the number of mixture components M and hidden states per component s. We ended up using $m = 15$ and $s = 10$. We then used the same values of m

and s for SpaMHMM and we did 3-fold cross-validation for the regularization hyperparameter λ in the range $[10^{-4}, 1]$. The value $\lambda = 10^{-1}$ was chosen. We also cross-validated the number of hidden states in 1-HMM and k-HMM around the values indicated in Section 2.3.3. Every model was trained for 100 iterations of EM or until the loss plateaus. For SpaMHMM, we did 100 iterations of the inner loop on each M-step, using a learning rate $\rho = 10^{-3}$. We repeat training 10 times for each model, starting from different random initializations, in order to reduce the likelihood of erroneous results due to local minima trapping.

Models were evaluated by computing the average log-likelihood per sample on normal and anomalous test data, plotting the receiver operating characteristic (ROC) curves and computing the respective areas under the curves (AUCs). The small standard deviations in Table 2.2 attest the robustness of the adopted initialization scheme and learning algorithms. Figure 2.6 shows that the ROC curves for MHMM and SpaMHMM are very similar and that these models clearly outperform 1-HMM and k-HMM. This is confirmed by the AUC and log-likelihood results in Table 2.2. Although k-HMM achieved the best (lowest) average log-likelihood on anomalous data, this result is not relevant, since it also achieved the worst (lowest) average log-likelihood on normal data. This is in fact the model with the worst performance, as shown by its ROC and respective AUC.

The bad performance of k-HMM likely results mostly from the small amount of data that each of the K models is trained with: in k-HMM, each HMM is trained with the data from the graph node (AP) that it is assigned to. The low log-likelihood value of the normal test data in this model confirms that the model does not generalize well to the test data and is probably highly biased towards the training data distribution. On the other hand, in 1-HMM there is a single HMM that is trained with the whole training set. However, the same HMM needs to capture the distribution of the data coming from all APs. Since each AP has its own typical usage profile, these data distributions are different and one single HMM may not be sufficiently expressive to learn all of them correctly. MHMM and SpaMHMM combine the advantages and avoid the disadvantages of both previous models. Clearly, since the mixtures for each node share the same dictionary of HMMs, every model in the mixture is trained with sequences from all graph nodes, at least in the first few training iterations. Thus, at this stage, the models can capture behaviors that are shared by all APs. As mixtures become sparser during training, some components in the dictionary can specialize in the distribution of a few APs. This avoids the problem

observed in the 1-HMM, which is unaware of the AP where a sequence originates. We would also expect SpaMHMM to be sparser and have better performance than MHMM, but only the former supposition was true (see Figure 2.7). The absence of performance gains in SpaMHMM might be explained by the fact that this dataset consists of simulated data, where users are static (i.e. they do not swap between APs unless the AP where they are connected stops working) and so the assumption that closer APs have similar distributions does not bring any advantage.

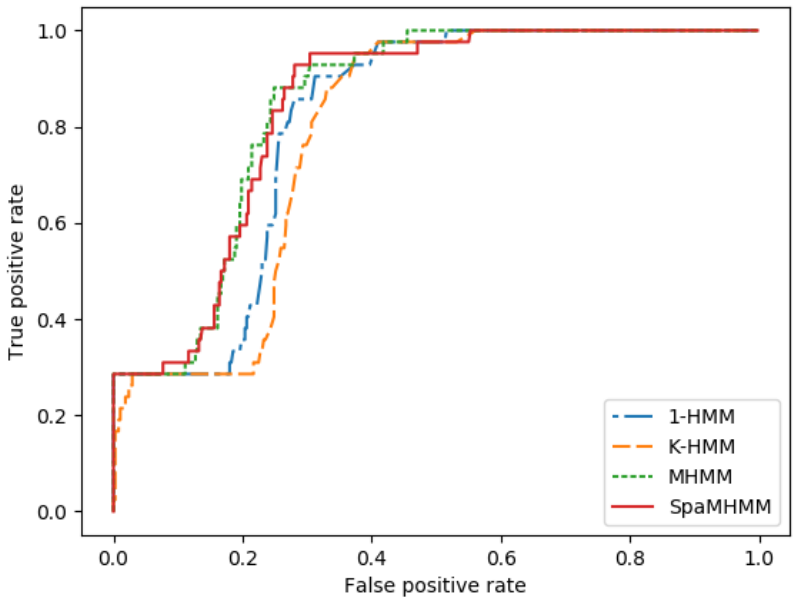


FIGURE 2.6: ROC curves for each model on the Wi-Fi dataset, for one of the 10 runs.

	AUC	Average log-likelihood Normal data	Anomalous data
1-HMM	0.806 (±0.01)	−6.36 (±0.66)	−129.40 (±22.22)
k-HMM	0.776 (±0.01)	−22.09 (±1.12)	**−130.36** (±26.30)
MHMM	**0.830** (±0.01)	−3.31 (±0.21)	−10.99 (±1.10)
SpaMHMM	0.829 (±0.01)	**−3.26** (±0.12)	−11.29 (±1.39)

TABLE 2.2: AUC and average log-likelihood per sample for each model in the Wi-Fi dataset averaged over 10 training runs. Standard deviations are in brackets. Best results are in bold.

2.3.5 Human motion forecasting

The human body is constituted by several interdependent parts, which interact as a whole producing sensible global motion patterns. These patterns may correspond to multiple activities like walking, eating, etc. Here, we use our model to make short-time prediction of sequences of human joint positions, represented as motion capture (mocap) data. The current state of the art methodologies use architectures based on deep recurrent neural networks (RNNs), achieving remarkable results both in short-time prediction (Fragkiadaki et al. [48], Martinez et al. [49]) and in long-term motion generation (Jain et al. [50], Pavllo et al. [51]).

Our experiments were conducted on the Human3.6M dataset from Ionescu et al. [52, 53], consisting of mocap data from 7 subjects performing 15 distinct actions. In this experiment, we have considered only 4 of those actions, namely "walking", "eating", "smoking" and "discussion". There, the human skeleton is represented with 32 joints whose position is recorded at 50 Hz. We build our 32x32-dimensional symmetric matrix $\boldsymbol{G}$ representing the graph $\mathcal{G}$ in the following sensible manner: $G_{y,y'} = 1$, if there is an actual skeleton connection between joints y and y' (e.g. the elbow joint is connected to the wrist joint by the forearm); $G_{y,y} = 1$, if joints y and y' are symmetric (e.g. left and right elbows); $G_{y,y'} = 0$, otherwise.

2.3.5.1 Forecasting

We reproduced as much as possible the experimental setup followed in Fragkiadaki et al. [48]. Specifically, we down-sampled the data by a factor of 2 and transformed the raw 3-D angles into an exponential map representation. We removed joints with constant exponential map, yielding a dataset with 22 distinct joints, and pruned our matrix $\boldsymbol{G}$ accordingly. Training was performed using data from 6 subjects, leaving one subject (denoted in the dataset by "S5") for testing. We did 3-fold cross-validation on the training data of the action "walking" to find the optimal number of mixture components m and hidden states s for the baseline mixture MHMM. Unsurprisingly, since this model can hardly overfit in such a complex task, we ended up with $m = 18$ and $s = 12$, which were the largest values in the ranges we defined. Larger values are likely to improve the results, but the training time would become too large to be practical. For SpaMHMM, we used these same values of m and s and we did 3-fold cross-validation on the training data of the action "walking" to fine-tune the value of λ in the range $[10^{-4}, 1]$. We ended up using $\lambda = 0.05$. The

number of hidden states in 1-HMM was set to 51 and in k-HMM it was set to 11 hidden states per HMM. The same values were then used to train the models for the remaining actions. Every model was trained for 100 iterations of EM or until the loss plateaus. For SpaMHMM, we did 100 iterations of the inner loop on each M-step, using a learning rate $\rho = 10^{-2}$.

In order to generate predictions for a joint (node) y starting from a given prefix sequence $\boldsymbol{X}_{\text{pref}}$, we compute the posterior distribution $p(\mathbf{X}|\boldsymbol{X}_{\text{pref}}, y)$ (see details in Section A.2) and we sample sequences from that posterior. Our evaluation method and metric again followed Fragkiadaki et al. [48]. We fed our model with 8 prefix subsequences with 50 frames each (corresponding to 2 seconds) for each joint from the test subject and we predicted the following 10 frames (corresponding to 400 miliseconds). Each prediction was built by sampling 100 sequences from the posterior and averaging. We then computed the average mean angle error for the 8 sequences at different time horizons.

Results are in Table 2.3. Among our models (1-HMM, k-HMM, MHMM, and Spa MHMM), SpaMHMM outperformed the remaining in all actions except "eating". For this action in particular, MHMM was slightly better than SpaMHMM, probably due to the lack of symmetry between the right and left sides of the body, which was one of the prior assumptions that we have used to build the graph $\mathcal{G}$. "Smoking" and "discussion" activities may also be highly non-symmetric, but results in our and others' models show that these activities are generally harder to predict than "walking" and "eating'. Thus, here, the skeleton structure information encoded in $\mathcal{G}$ behaves as a useful prior for SpaMHMM, guiding it towards better solutions than MHMM. The worse results for 1-HMM and K-HMM likely result from the same limitations that we have pointed out in Section 2.3.4: each component in k-HMM is inherently trained with less data than the remaining models, while 1-HMM does not make distinction between different graph nodes. Extending the discussion to the state of the art solutions for this problem, we note that SpaMHMM compares favorably with ERD, LSTM-3LR and SRNN, which are all RNN-based architectures. Moreover, ERD and LSTM-3LR were explicitly designed for this task, which is not the case of SpaMHMM. This is also true for GRU supervised and QuaterNet, which clearly outperform all remaining models, including ours. This is unsurprising, since RNNs are capable of modeling more complex dynamics than HMMs, due to their intrinsic non-linearity and continuous state representation. This also allows their usage for long-term

motion generation, in which HMMs do not behave well due to their finite and discrete dynamics and lack of long-term memory. However, unlike GRU supervised and QuaterNet, SpaMHMM models the probability distribution of the data directly, allowing its application in domains like novelty detection. Regarding sparsity, the experiments confirm that the SpaMHMM mixture coefficients are actually sparser than those of MHMM, as shown in Figure 2.7.

	Walking				Eating				Smoking				Discussion			
miliseconds	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
1-HMM	0.91	1.04	1.22	1.31	1.00	1.08	1.15	1.21	1.45	1.55	1.70	1.75	1.19	1.42	1.55	1.56
k-HMM	1.29	1.33	1.34	1.38	1.16	1.22	1.28	1.34	1.70	1.77	1.90	1.95	1.47	1.61	1.68	1.63
MHMM	**0.78**	**0.93**	1.13	1.21	**0.77**	**0.87**	**0.98**	**1.06**	1.44	1.53	1.69	1.77	1.14	1.36	1.52	1.54
SpaMHMM	0.80	**0.93**	**1.11**	**1.18**	0.81	0.90	0.99	**1.06**	**1.29**	**1.39**	**1.61**	**1.67**	**1.09**	**1.30**	**1.44**	**1.49**
ERD [48]	0.93	1.18	1.59	1.78	1.27	1.45	1.66	1.80	1.66	1.95	2.35	2.42	2.27	2.47	2.68	2.76
LSTM-3LR [48]	0.77	1.00	1.29	1.47	0.89	1.09	1.35	1.46	1.34	1.65	2.04	2.16	1.88	2.12	2.25	2.23
SRNN [50]	0.81	0.94	1.16	1.30	0.97	1.14	1.35	1.46	1.45	1.68	1.94	2.08	1.22	1.49	1.83	1.93
GRU sup. [49]	0.28	0.49	0.72	0.81	0.23	0.39	0.62	0.76	0.33	0.61	1.05	1.15	0.31	0.68	1.01	1.09
QuaterNet [51]	0.21	0.34	0.56	0.62	0.20	0.35	0.58	0.70	0.25	0.47	0.93	0.90	0.26	0.60	0.85	0.93

TABLE 2.3: Mean angle error for short-term motion prediction on Human3.6M for different actions and time horizons. The results for ERD, LSTM-3LR, SRNN, GRU supervised and QuaterNet were extracted from Pavllo et al. [51]. Best results among our models are in bold, best overall results are underlined.

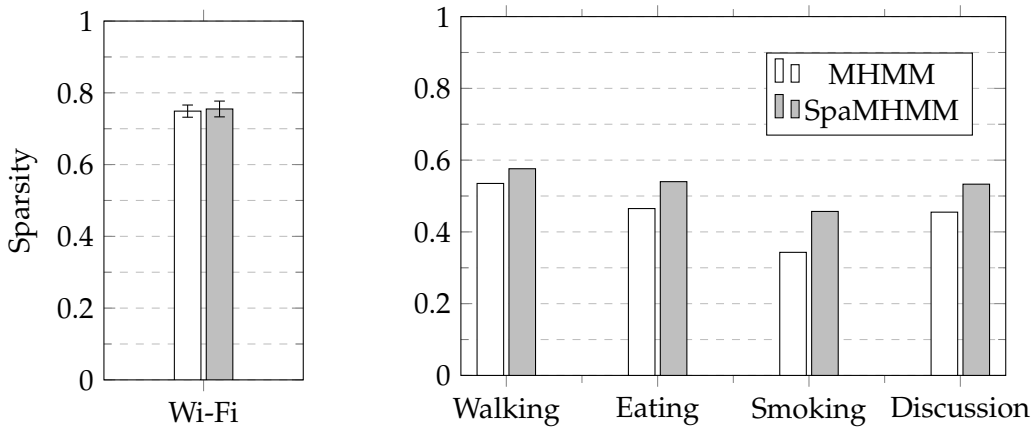


FIGURE 2.7: Relative sparsity (number of coefficients equal to zero / total number of coefficients) of the obtained MHMM and SpaMHMM models on the Wi-Fi dataset (left) and on the Human3.6M dataset for different actions (right). For the Wi-Fi dataset, the average value over the 10 training runs is shown together with the standard deviation. Both models for the Wi-Fi dataset have 150 coefficients. All models for the Human3.6M dataset have 396 coefficients.

2.3.5.2 Joint cluster analysis

We may roughly divide the human body in four distinct parts: upper body (head, neck and shoulders), arms, torso and legs. Joints that belong to the same part naturally tend to have coherent motion, so we would expect them to be described by more or less the

same components in our mixture models (MHMM and SpaMHMM). Since SpaMHMM is trained to exploit the known skeleton structure, this effect should be even more apparent in SpaMHMM than in MHMM. To confirm this conjecture, we have trained MHMM and SpaMHMM for the action "walking" using four mixture components only, i.e. $m = 4$, and we have looked for the most likely component (cluster) for each joint:

$$C_y \triangleq \underset{z\in\{1,\dots,m\}}{\arg\max}\, p(z \mid y) = \underset{z\in\{1,\dots,m\}}{\arg\max}\, \alpha_z^{(y)}, \tag{2.25}$$

where C_y is, therefore, the cluster assigned to joint y. The results are in Figure 2.8. From there we can see that MHMM somehow succeeds on dividing the body in two main parts, by assigning the joints in the torso and in the upper body mostly to the red/'+' cluster, while those in the hips, legs and feet are almost all assigned to the green/'△' cluster. Besides, we see that in the vast majority of the cases, symmetric joints are assigned to the same cluster. These observations confirm that we have chosen the graph $\mathcal{G}$ for this problem in an appropriate manner. However, some assignments are unnatural: e.g. one of the joints in the left foot is assigned to the red/'+' cluster and the blue/'○' cluster is assigned to one single joint, in the left forearm. We also observe that the distribution of joints per clusters is highly uneven, being the green/'△' cluster the most represented by far. SpaMHMM, on the other hand, succeeds on dividing the body in four meaningful regions: upper body and upper spine in the green/'△' cluster; arms in the blue/'○' cluster; lower spine and hips in the orange/'x' cluster; legs and feet in the red/'+' cluster. Note that the graph $\mathcal{G}$ used to regularize SpaMHMM does not include any information about the body part that a joint belongs to, but only about the joints that connect to it and that are symmetric to it. Nevertheless, the model is capable of using this information together with the training data in order to divide the skeleton in an intuitive and natural way. Moreover, the distribution of joints per cluster is much more even in this case, which may also help to explain why SpaMHMM outperforms MHMM: by splitting the joints more or less evenly by the different HMMs in the mixture, none of the HMM components is forced to learn too many motion patterns. In MHMM, we see that the green/'+' component, for instance, is the most responsible for modeling the motion of almost all joints in the legs and hips and also some joints in the arms and the red/'+' component is prevalent in the prediction of the motion patterns of the neck and left foot, which are presumably very different.

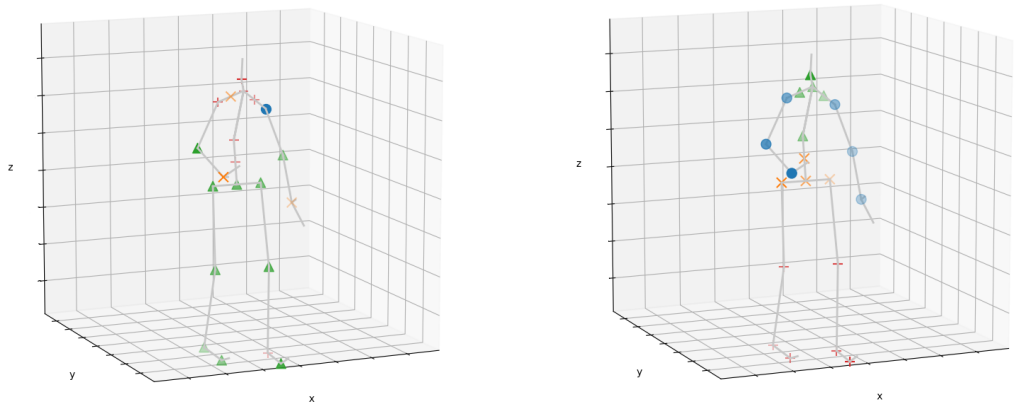


FIGURE 2.8: Assignments of joints to clusters in MHMM (left) and SpaMHMM (right). The different colors (blue, green, orange, red) and the respective symbols ('○', '△', 'x', '+') on each joint represent the cluster that the joint was assigned to. on each joint represent the cluster that the joint was assigned to.

2.3.6 Conclusion

We proposed a method to model the generative distribution of sequential data from nodes connected in a graph with a known fixed topology. The method is based on a mixture of HMMs where its coefficients are regularized during the learning process in such a way that affine nodes will tend to have similar coefficients, exploiting the known graph structure. We also prove that pairwise optimization of the coefficients leads to sparse mixtures. Experimental results suggest that sparsity holds in the general case. We evaluate the method performance in two completely different tasks (anomaly detection in Wi-Fi networks and human motion forecasting), showing its effectiveness and versatility.

2.4 Summary and directions for future work

After a detailed presentation of SOHMMM and SpaMHMM algorithms in sections 2.2 and 2.3, respectively, it is instructive to offer a brief comparative overview of the two models. Additionally, we shall discuss how these frameworks could be extended and generalized.

2.4.1 SOHMMM vs. SpaMHMM

The similarities between SOHMMM and SpaMHMM go beyond the obvious fact that both models use the hidden Markov model as their core component. It should be noted that equations (2.3) and (2.18), which govern the learning dynamics of these models, are both structurally similar to equation (2.1) and, therefore, similar to each other. Despite the

fact that the former is an energy function and the latter is a proper density, the SOHMMM energy (2.3) could also be transformed into a density by normalization:

$$p(\mathbf{X};\Theta) = \frac{E(\mathbf{X};\Theta)}{Z(\Theta)}, \quad \text{where} \quad Z(\Theta) = \int E(\mathbf{X};\Theta)\, \mathrm{d}\mathbf{X}. \tag{2.26}$$

This observation makes it clear that, in the learning stage, SOHMMM can also be regarded as a mixture of HMMs.

However, major differences arise when we examine how the two models associate entities y with atoms z. SOHMMM does not include any explicit dependency on y since there is a one-to-one correspondence between entities and atoms in the lattice (e.g. in the experiment with wireless simulation data, each AP is modeled by one HMM and there are not two APs sharing the same HMM). Moreover, for the mixture model to exist, the topology of the lattice must be known or at least a distance must be defined between each pair of nodes. On the other hand, in SpaMHMM, the mixture is defined by an explicit many-to-many relationship between entities and atoms, defined by the distribution $p(\mathrm{z} \mid \mathrm{y})$. As a result, the model for each entity is a mixture of HMMs and nothing impedes two distinct entities of sharing the same HMM, although possibly assigned to different mixture weights. Thus, SpaMHMM is a more expressive model than SOHMMM, which comes at the cost of having a few extra parameters to model $p(\mathrm{z} \mid \mathrm{y})$. This would also allow new entities to be added to the model in a simple manner and without affecting the remaining entities' models, as will be discussed in Section 2.4.5. Additionally, in SpaMHMM, the graph structure is used as a regularization term only, so, in case this information is absent, the model can still be learned and the correlations between multiple entities would still be exploited, although no prior information about them would be provided to the learning algorithm.

Another apparent difference between the two models is the fact that SOHMMM was conceived to be learned online and SpaMHMM was presented with an offline learning algorithm. These were mere design choices, though. For instance, it is straightforward to replace SOHMMM stochastic gradient descent (SGD) algorithm with a batch version and SGD for SpaMHMM could also be derived following the same principles that were used in its derivation for SOHMMM.

2.4.2 Generalizing SpaMHMM

Two obvious limitations of SpaMHMM are its inherent dependency on hidden Markov models, which have limited expressiveness, and the fact that its learning algorithm was conceived to an offline and non-distributed setting. In this section, we discuss how these two drawbacks could be overcome by an extended and generalized framework, of which SpaMHMM is a particular case. Specifically, we consider again the setting where each entity y in a set $\mathcal{Y}$ produces sequences $\mathbf{X} = \left(\mathbf{x}^{(1)}, ..., \mathbf{x}^{(T)}\right)$. The prior knowledge about the pairwise affinity degrees between entities in $\mathcal{Y}$ at time t is summarized in an undirected weighted graph $\mathcal{G}$, where each entity corresponds to a graph node. The ultimate goal is to find the distribution of the observed sequences given the corresponding node, that is, to model the generative distribution $p(\mathbf{X} \mid \mathrm{y})$. For this purpose, we shall consider models of the form:

$$p(\mathbf{X} \mid \mathrm{y}) = \int p(\mathbf{X} \mid \mathbf{z}) p(\mathbf{z} \mid \mathrm{y}) \, \mathrm{d}\mathbf{z}. \tag{2.27}$$

This model comprises a latent space that is shared by all entities and where all observations $\mathbf{X}$ are produced, according to a *conditional observation density* $p(\mathbf{X} \mid \mathrm{z})$. Through the *latent density* $p(\mathrm{z} \mid \mathrm{y})$, each entity chooses regions of the latent space that are more likely to explain its own sequences. While not specifying the structure of both $p(\mathbf{X} \mid \mathrm{z})$ and $p(\mathrm{z} \mid \mathrm{y})$, this is a meta-model that makes no assumptions on the nature of the observed data streams. However, depending on those choices, exact inference on the resulting model may or may not be a tractable problem.

Clearly, the SpaMHMM model is a particular case of equation (2.27), where the latent space is discrete and finite and the latent generative model is an HMM parametrized by $\mathbf{z}$. The discrete nature of the latent space and the underlying independence assumptions of the HMM allow for efficient exact inference in the resulting model using Algorithm 1.1.

An interesting, more expressive, and still unexplored possibility would be to implement the conditional observation model $p(\mathbf{X} \mid \mathrm{z})$ through a recurrent mixture density network (RMDN) [54–56]. An RMDN consists of a recurrent neural network (RNN) whose outputs at each time step parametrize a Gaussian mixture model with c components:

$$p(\boldsymbol{X} \mid \boldsymbol{z}) = \prod_{t=1}^{T} p(\boldsymbol{x}^{(t)} \mid \boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(t-1)}, \boldsymbol{z}) = \prod_{t=1}^{T} \sum_{j=1}^{c} \alpha_j^{(t)} \mathcal{N}(\boldsymbol{x}^{(t)}; \boldsymbol{\mu}_j^{(t)}, \Sigma_j^{(t)}), \tag{2.28}$$

$$\{\alpha_j^{(t)}, \boldsymbol{\mu}_j^{(t)}, \Sigma_j^{(t)}\}_{j=1}^{c} = \mathrm{RNN}(\mathbf{x}^{(1)}, ..., \mathbf{x}^{(t-1)}; \boldsymbol{z}). \tag{2.29}$$

The latent space may be discrete, in which case we shall end up with a discrete mixture of RMDNs, where exact inference is straightforward. Another possibility is to choose a continuous latent space and, in such scenario, a sensible option is using a Gaussian density for the latent density model:

$$p(z \mid y) = \mathcal{N}(z; \boldsymbol{\mu}_y, \Sigma_y). \tag{2.30}$$

Under this setting, computing equation (2.27) becomes infeasible and, hence, exact inference is not possible. However, the marginals $p(\boldsymbol{X} \mid y)$ may be approximated using a Monte Carlo estimation:

$$p(\boldsymbol{X} \mid y) \approx \frac{1}{n} \sum_{i=1}^{n} p(\boldsymbol{X} \mid z_i), \tag{2.31}$$

where the z_i are sampled from $p(\mathbf{z} \mid y)$ and n is the desired number of samples, which should be sufficiently large for the estimator to have a small variance.

Although very different in terms of their internal structure, all aforementioned models share the external structure defined by equation (2.27). In the forthcoming sections, we outline how training and inference could be done in this broad family of models under multiple settings.

2.4.3 Offline, centralized learning

It is reasonable to assume that an initial dataset composed of sequences from all entities may be gathered and used to learn an initial model for all entities. This model could then be refined online in each instance using a distributed algorithm. Hence, it makes sense to derive a algorithm to learn the meta-model defined by equation (2.27) in an offline and centralized setting. This is the idea we followed in Section 2.3, which solves this problem for the particular of the SpaMHMM model using EM. This algorithm is based on the alternated maximization over variational densities $q(\mathbf{z})$ and model parameters of an evidence lower bound (ELBO):

$$\log p(\mathbf{X} \mid \mathrm{y}) \geq \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z})} \left[\log p(\mathbf{X}, \mathbf{z} \mid \mathrm{y}) \right] - \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z})} \left[\log q(\mathbf{z}) \right] \triangleq \mathrm{ELBO}. \tag{2.32}$$

For fixed model parameters, maximization of the ELBO with respect to $q(\mathbf{z})$ is attained when this distribution matches the true posterior $p(\mathbf{z} \mid \mathbf{X}, \mathrm{y})$. For the case of SpaMHMM, computing this posterior is a tractable problem. When this is not the case, some options

are constraining the variational density to have a simpler, factorizable form (e.g. mean-field approximation, [57]) or approximating the posterior with a parametric model, like a deep neural network (e.g. variational auto-encoders [9]).

In any of those cases, leveraging the prior contextual information represented by the graph $\mathcal{G}$ through the introduction of a regularizer term like in equation (2.21) poses no significant additional difficulties to the optimization algorithm.

2.4.4 Online, centralized learning

Assuming that a few new training examples $(\boldsymbol{X}_i, y_i)$ are periodically uploaded to a central unit and that entities are able to download updated model parameters from this unit, it is a reasonable goal to be able to update the conditional observation model $p(\mathbf{X} \mid \mathbf{z})$ and the latent density model $p(\mathbf{z} \mid \mathrm{y})$.

Incremental training of HMMs has been widely treated in literature. Existing approaches fall into one of three categories: direct gradient-based optimization over the log-likelihood objective (Baldi and Chauvin [28]), incremental versions of EM (Digalakis [58], Mongillo and Deneve [59]), and recursive maximum likelihood estimation (Rydén [60]). Khreich et al. [35] provided an overview of these methods. Both gradient-based optimization and incremental EM can be applied to SpaMHMM, making it suitable for an online learning setting. More complex models, particularly those based on neural networks, are typically trained using stochastic gradient descent (or any of its many variants), which is inherently applicable in an online setting. If an updated graph $\mathcal{G}$ is available, this information can also be incorporated into the learning objective as before.

2.4.5 Online, distributed learning

We now consider a scenario where each entity aims to update its own model autonomously, that is, without observing any information from the remaining entities. Therefore, we assume y is fixed to some $y \in \mathcal{Y}$ and, therefore, all new training examples are of the form $(\boldsymbol{X}_i, y)$.

This particular setting is favorably accommodated by the structure of equation (2.27). By comprising a shared (and desirably rich) conditional observation model $p(\mathbf{X} \mid \mathrm{z})$ and a (presumably simpler) entity-dependent latent density model $p(\mathrm{z} \mid \mathrm{y})$, an entity y can adapt its own model $p(\mathbf{X} \mid y)$ without changing the model $p(\mathbf{X} \mid y)$ for any other entity $y' \neq y$. The idea here is to freeze the parameters of the conditional observation model and

update only the latent density model $p(\mathrm{z} \mid y)$ to accommodate the new data. By doing this, the entity y is performing a search over the latent space to find regions where the new observations have a higher likelihood. If the resulting model is still unsatisfactory, this likely means that the shared latent space was still not diverse enough to explain the new data. In this circumstance, the entity may decide to upload its observations $(\boldsymbol{X}_i, y)$ to a central unit, which in turn may update the whole model as outlined in Section 2.4.4.

Chapter 3

Multi-source domain adaptation

Some parts of this chapter were originally published in or adapted from:

> [61] F. T. de Andrade, "Adversarial domain adaptation for sensor networks," Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2020 (presented in Section 3.3)
>
> [62] D. Pernes and J. S. Cardoso, "Tackling unsupervised multi-source domain adaptation with optimism and consistency," *CoRR*, vol. abs/2009.13939, 2020 (presented in Section 3.4)

The Master's thesis [61] was supervised by Jaime S. Cardoso and co-supervised by Diogo Pernes.

3.1 Introduction

In Chapter 2, we have addressed the situation where the set of entities was the same at training and testing time. The goal there was to exploit inter-correlations between entities to learn better generative models for each individual entity. Now, we focus on the problem of learning a discriminative model for one particular entity (the *target*) for which no annotated data is available. Assuming some invariance properties, we can hope to accomplish this task by learning a discriminative model using the combination of annotated data from the remaining entities (the *sources*) and unlabeled data from the target entity. Since entities do not need to (and generally do not) correspond to physical objects and may refer to different contexts where the data was collected, they are more commonly called *domains* and the problem itself is known as *domain adaptation* (DA). In the following, we motivate the practical importance of this problem and summarize our contributions.

Supervised training of deep neural networks has achieved outstanding results on multiple learning tasks greatly due to the availability of large and rich annotated datasets. Unfortunately, annotating such large-scale datasets is often prohibitively time-consuming and expensive. Furthermore, in many practical cases, it is not possible to collect annotated data with the same characteristics as the test data, and, as a result, training and test data are drawn from distinct underlying distributions. As a consequence, the model performance tends to decrease significantly on the test data. The goal of DA algorithms is to minimize this gap by finding transferable knowledge from the source to the target domain. Sometimes, it is assumed that a small portion of labeled target data are available at training time – a setting that is known as *semi-supervised* DA (e.g. Daumé III et al. [63], Donahue et al. [64], Kumar et al. [65], Saito et al. [66], Yao et al. [67]). In this chapter, we focus mostly on the more challenging scenario, where no labeled target data are available for training – known as *unsupervised* DA (e.g. Baktashmotlagh et al. [68], Ganin and Lempitsky [69], Kang et al. [70], Long et al. [71], Zhao et al. [72]). The DA problem, in its semi-supervised and unsupervised variants, has received increased attention in recent years, both from theoretical (e.g. Ben-David et al. [73, 74], Blitzer et al. [75], Cortes and Mohri [76], Gopalan et al. [77], Hoffman et al. [78], Zhao et al. [79]) and algorithmic perspectives (e.g. Ajakan et al. [80], Becker et al. [81], Fernando et al. [82], Jhuo et al. [83], Long et al. [84], Louizos et al. [85], Sun et al. [86], Tzeng et al. [87]). In many situations, the annotated training data may consist of a combination of distinct datasets, some of which may be closer or farther away from the target data. Finding nontrivial ways of combining multiple datasets to approximate the target distribution and extracting relevant knowledge from such combination is the purpose of multi-source DA algorithms (e.g. Zhao et al. [72], Hoffman et al. [78], Kim et al. [88], Guo et al. [89], Mansour et al. [90], Schoenauer-Sebag et al. [91], Zhang et al. [92]) and is also our main focus in this chapter.

The remainder of the chapter is organized as follows: i) we formalize the problem and provide some useful background by reviewing some important theoretical results and state of the art algorithms (Section 3.2); ii) we discuss some exploratory solutions for this problem in the context of sensor networks (Section 3.3); and finally iii) we present our own novel algorithm for unsupervised multi-source domain adaptation (Section 3.4).

3.2 Background

We start by analyzing the problem of DA from a theoretical perspective and then we overview the main approaches that constitute the state of the art. Although some authors have considered the problem of DA for regression problems (e.g. Cortes and Mohri [93], Zhao et al. [72]), the literature on classification is far more vast. Moreover, many of the results and methods explained in this section can be extended to regression problems with minor modifications. For these reasons, we shall focus our discussion mostly on classification.

3.2.1 Theoretical foundation

3.2.1.1 Single source setting

Ben-David et al. [73] developed a rigorous yet comprehensive theoretical model for domain adaptation that we summarize here. This formulation enlightens the intrinsic difficulties associated with this task and provides a deep foundation to many of the algorithms we discuss in this chapter and, particularly, to our own approach, presented in Section 3.4.

Before we present and discuss the most important results, let us introduce a few preliminary definitions. A *domain* $\mathcal{D}$ is defined by a joint distribution $p_{\mathcal{D}}(\mathbf{x}, \mathbf{y})$ over input features $\mathbf{x} \in \mathcal{X}$ and target variables $\mathbf{y} \in \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ denote the input and target spaces, respectively. For the domain adaptation task to be well defined, at least two domains must be considered: a *source* domain $\mathcal{S}$, with joint distribution denoted by $p_{\mathcal{S}}(\mathbf{x}, \mathbf{y})$, from which abundant annotated data is usually available, and a *target* domain $\mathcal{T}$, with joint distribution $p_{\mathcal{T}}(\mathbf{x}, \mathbf{y})$, from which scarce or even zero annotated data is available at training time. Following most classical results from statistical learning theory, Ben-David et al. [73] focused on binary classification, thus $\mathcal{Y} = \{0, 1\}$. Under this setting, it is possible to define a *labeling function* $f_{\mathcal{D}} : \mathcal{X} \mapsto [0, 1]$ for each domain, given by $f_{\mathcal{D}}(\boldsymbol{x}) = p_{\mathcal{D}}(\mathrm{y} = 1 \mid \boldsymbol{x})$. A *hypothesis* is any function $h : \mathcal{X} \mapsto \{0, 1\}$ and a set $\mathcal{H}$ of these functions is called a *hypothesis class*. The expected absolute difference between h and $f_{\mathcal{D}}$ is called the *risk* (or *error*) of hypothesis h (with respect to the labeling function $f_{\mathcal{D}}$):

$$\epsilon(h, f_{\mathcal{D}}) \triangleq \mathbb{E}_{\mathbf{x} \sim p_{\mathcal{D}}} \left[|h(\mathbf{x}) - f_{\mathcal{D}}(\mathbf{x})| \right]. \tag{3.1}$$

We use $\epsilon_{\mathcal{S}}(h)$ and $\epsilon_{\mathcal{T}}(h)$ as shorthands for $\epsilon(h, f_{\mathcal{S}})$ and $\epsilon(h, f_{\mathcal{T}})$ and refer to them as the source and target risks (or errors), respectively. The empirical estimates of these are denoted as $\widehat{\epsilon}_{\mathcal{S}}(h)$ and $\widehat{\epsilon}_{\mathcal{T}}(h)$, respectively.

Given two domains $\mathcal{D}$ and $\mathcal{D}'$ and a hypothesis class $\mathcal{H}$, the $\mathcal{H}$-divergence provides a distance measure between the marginal distributions of features in $\mathcal{D}$ and $\mathcal{D}'$ (according to $\mathcal{H}$):

$$d_{\mathcal{H}}(\mathcal{D}, \mathcal{D}') \triangleq \sup_{h \in \mathcal{H}} 2|\Pr_{\mathcal{D}}(\mathbf{1}_h) - \Pr_{\mathcal{D}'}(\mathbf{1}_h)|,$$

where $\mathbf{1}_h \triangleq \{\boldsymbol{x} \in \mathcal{X} : h(\boldsymbol{x}) = 1\}$ and $\Pr_{\mathcal{D}}(\mathbf{1}_h)$ is the probability assigned by the density $p_{\mathcal{D}}(\mathbf{x})$ to the subset $\mathbf{1}_h \subseteq \mathcal{X}$. As is often the case, when the true underlying marginal distributions are unknown or intractable but finite sets of (unlabeled) samples from both domains are available, an empirical $\mathcal{H}$-divergence can be constructed by replacing the true probabilities $\Pr_{\mathcal{D}}(\mathbf{1}_h)$ and $\Pr_{\mathcal{D}'}(\mathbf{1}_h)$ by the respective empirical estimates. Remarkably, under weak conditions on the hypothesis class, computing this empirical $\mathcal{H}$-divergence is equivalent to finding the hypothesis in $\mathcal{H}$ that maximally discriminates between samples of the two domains. This result is enunciated formally in Lemma 3.1 and, as we shall see later, is exploited by adversarial-based approaches for domain adaptation.

Lemma 3.1. *(Lemma 2 from Ben-David et al. [73]) Let $\mathcal{H}$ be a hypothesis class such that if $h \in \mathcal{H}$ then $1 - h \in \mathcal{H}$. Given two sets $\widehat{\mathcal{D}}$ and $\widehat{\mathcal{D}}'$ of n samples each drawn from two domains D and D', the empirical $\mathcal{H}$-divergence between $\mathcal{D}$ and $\mathcal{D}'$ is given by:*

$$\widehat{d}_{\mathcal{H}}(\mathcal{D}, \mathcal{D}') = 2\left(1 - \min_{h \in \mathcal{H}}\left[\frac{1}{n}\sum_{\boldsymbol{x}:\, h(\boldsymbol{x})=1} \mathbf{1}_{\boldsymbol{x} \in \widehat{\mathcal{D}}} + \frac{1}{n}\sum_{\boldsymbol{x}:\, h(\boldsymbol{x})=0} \mathbf{1}_{\boldsymbol{x} \in \widehat{\mathcal{D}}'}\right]\right). \tag{3.2}$$

Note that if the two sets $\widehat{\mathcal{D}}$ and $\widehat{\mathcal{D}}'$ can be discriminated perfectly by a hypothesis $h \in \mathcal{H}$ (i.e. if there is an $h \in \mathcal{H}$ such that $h(\boldsymbol{x}) = 0$ if $x \in \widehat{\mathcal{D}}$ and $h(\boldsymbol{x}) = 1$ if $x \in \widehat{\mathcal{D}}'$), then $\widehat{d}_{\mathcal{H}}(\mathcal{D}, \mathcal{D}')$ is maximum and equal to 2.

For any hypothesis class $\mathcal{H}$, we may define the *symmetric difference hypothesis class* $\mathcal{H}\Delta\mathcal{H}$ as:

$$\mathcal{H}\Delta\mathcal{H} \triangleq \{l : l(\boldsymbol{x}) = h(\boldsymbol{x}) \oplus h'(\boldsymbol{x}),\ h, h' \in \mathcal{H}\}, \tag{3.3}$$

where $\oplus$ denotes the "exclusive or" (xor) operation. Combining this definition with equation (3.2.1.1), the definition of $\mathcal{H}\Delta\mathcal{H}$-divergence, which happens to play a major role in

Theorem 3.2, follows immediately. This theorem is the main result in this section as it provides an upper bound for the target risk given the source risk and the $\mathcal{H}\Delta\mathcal{H}$-divergence between the source and target domains.

Theorem 3.2. *(Theorem 2 from Ben-David et al. [73]) Let $\mathcal{H}$ be a hypothesis class with VC-dimension d. Consider n unlabeled samples drawn from each of the two domains $\mathcal{S}$ (source) and $\mathcal{T}$ (target). Then, for every $h \in \mathcal{H}$ and any $\delta \in (0,1)$, with probability at least $1-\delta$ over the choice of samples,*

$$\epsilon_{\mathcal{T}}(h) \leq \epsilon_{\mathcal{S}}(h) + \frac{1}{2}\widehat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T}) + \lambda + 2\sqrt{\frac{2d\log(2n) + \log(\frac{2}{\delta})}{n}}, \tag{3.4}$$

where $\lambda \triangleq \min_{h\in\mathcal{H}} \epsilon_{\mathcal{S}}(h) + \epsilon_{\mathcal{T}}(h)$.

This bound immediately confirms the intuition that a low target error can be achieved by training a classifier to minimize the error in the source domain, provided that the marginal distributions of features are similar (i.e. $\widehat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T})$ is small) and a low error on the combination of the two domains can be achieved (i.e. λ is also small). A deeper and more complete interpretation of this bound shall be provided later on, when we take into account the fact that, by applying deep neural networks, we can not only construct rich hypothesis classes but also manipulate and learn feature representations. The latter observation suggests that this kind of classifiers may also have an impact on the $\widehat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T})$ and λ terms in equation (3.4), which will indeed be the case.

3.2.1.2 Multi-source setting

So far we have only considered the setting where a single source domain was available. However, in many practical cases, the annotated training dataset consists of a collection of subdatasets, each one belonging to its own domain. Therefore, it makes sense to consider k distinct source domains $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ and, in particular, to see how Theorem 3.2 can be generalized to this setting.

Theorem 3.3. *(Theorem 2 from Zhao et al. [72]) Let $\mathcal{H}$ be a hypothesis class with VC-dimension d. Consider n unlabeled samples drawn from the target domain $\mathcal{T}$ and n/k annotated samples drawn from each of the k source domains $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$. Then, for every $h \in \mathcal{H}$, any $\boldsymbol{\alpha} \in [0,1]^k$: $\sum_{j=1}^k \alpha_i = 1$, and any $\delta \in (0,1)$, with probability at least $1-\delta$ over the choice of samples,*

$$\epsilon_{\mathcal{T}}(h) \leq \sum_{j=1}^{k} \alpha_j \left(\widehat{\epsilon}_{\mathcal{S}_j}(h) + \frac{1}{2}\widehat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}_j,\mathcal{T})\right) + \lambda_{\boldsymbol{\alpha}} + O\left(\sqrt{\frac{1}{n}\left(\log\frac{1}{\delta} + d\log\frac{n}{d}\right)}\right), \tag{3.5}$$

where $\lambda_{\boldsymbol{\alpha}} \triangleq \min_{h \in \mathcal{H}} \epsilon_{\mathcal{T}}(h) + \sum_{j=1}^{k} \alpha_j \epsilon_{\mathcal{S}_j}(h)$.

Unsurprisingly, the bound in Theorem 3.3 is essentially a convex combination of the bounds provided by Theorem 3.2 for each individual source domain. Thus, the same interpretation applies here. Nonetheless, the source weights $\boldsymbol{\alpha}$ provide an extra degree of freedom that should be taken into account. Depending on how much each source domain differs from the target, it may be beneficial to weight each source domain differently. Adjusting these weights is therefore an extra non-trivial task, exclusive to the multi-source setting, that may have a significant impact in the performance of the domain adaptation algorithm.

3.2.2 State of the art

We now do a brief overview of the most relevant DA algorithms, both in the single source and multi-source settings.

As we have just seen from a theoretical point of view, the success of the DA task depends on how similar the target domain is to the source(s), which is equivalent to saying that some properties of the underlying distributions must be invariant across domains. Different DA algorithms can therefore be categorized according to the invariance properties they assume.

3.2.2.1 Target shift

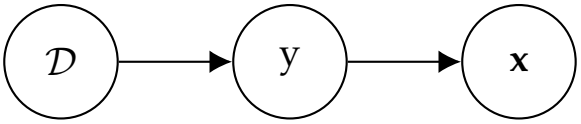


FIGURE 3.1: Graphical representation of the target shift setting as a Bayesian network.

In the *target shift* setting, only the marginal distribution of labels is allowed to vary across domains. A practical example where this assumption may be realistic is when the label y represents having or not a certain disease and the input features $\mathbf{x}$ are symptoms. It is plausible to assume that the prevalence of the disease may vary over time or across different populations, but the probability of some symptom being present or absent given that one has or not the disease should remain constant. Thus, as implied by Figure 3.1, the joint distribution of any domain $\mathcal{D}$ is assumed to factorize as $p_{\mathcal{D}}(\mathbf{x}, \mathrm{y}) = p(\mathbf{x} \mid \mathrm{y}) p_{\mathcal{D}}(\mathrm{y})$, where $p(\mathbf{x} \mid \mathrm{y})$ is domain-invariant. By further assuming that $\mathrm{Supp}(p_{\mathcal{T}}(\mathrm{y})) \subseteq$

$\text{Supp}(p_{\mathcal{S}}(\mathrm{y}))$, we have:

$$\begin{aligned} p_{\mathcal{T}}(\mathrm{y} \mid \mathbf{x}) &\propto p(\mathbf{x} \mid \mathrm{y}) p_{\mathcal{T}}(\mathrm{y}) \\ &= p(\mathbf{x} \mid \mathrm{y}) p_{\mathcal{S}}(\mathrm{y}) \frac{p_{\mathcal{T}}(\mathrm{y})}{p_{\mathcal{S}}(\mathrm{y})} \\ &\propto p_{\mathcal{S}}(\mathrm{y} \mid \mathbf{x}) \frac{p_{\mathcal{T}}(\mathrm{y})}{p_{\mathcal{S}}(\mathrm{y})}. \end{aligned} \tag{3.6}$$

Thus, if the class ratios $p_{\mathcal{T}}(\mathrm{y})/p_{\mathcal{S}}(\mathrm{y})$ are known, the problem of DA under target shift is solved by learning a probabilistic classifier on the source domain, reweighting it with the class ratios, and then normalizing the class scores. Therefore, the literature for DA under target shift focuses on estimating class ratios when these are unknown and cannot be estimated directly from the training data, due to the absence of labels for the target samples.

An elegant and simple solution to this problem was proposed by Lipton et al. [94]. Specifically, for any classifier $h : \mathcal{X} \mapsto \mathcal{Y}$ trained with labeled data from the source domain, the target shift assumption implies that $p_{\mathcal{T}}(h(\mathbf{x}) \mid \mathrm{y}) = p_{\mathcal{S}}(h(\mathbf{x}) \mid \mathrm{y})$ and hence:

$$\begin{aligned} p_{\mathcal{T}}(h(\mathbf{x})) &= \sum_{\mathrm{y}} p_{\mathcal{T}}(h(\mathbf{x}) \mid \mathrm{y}) p_{\mathcal{T}}(\mathrm{y}) \\ &= \sum_{\mathrm{y}} p_{\mathcal{S}}(h(\mathbf{x}) \mid \mathrm{y}) p_{\mathcal{T}}(\mathrm{y}) \qquad (3.7) \\ &= \sum_{\mathrm{y}} p_{\mathcal{S}}(h(\mathbf{x}), \mathrm{y}) \frac{p_{\mathcal{T}}(\mathrm{y})}{p_{\mathcal{S}}(\mathrm{y})}. \qquad (3.8) \end{aligned}$$

Note that $p_{\mathcal{S}}(h(\mathbf{x}) \mid \mathrm{y})$, $p_{\mathcal{S}}(h(\mathbf{x}), \mathrm{y})$, and $p_{\mathcal{S}}(\mathrm{y})$ can all be estimated from labeled source samples and $p_{\mathcal{T}}(h(\mathbf{x}))$ can be estimated from unlabeled target samples. Thus, one can either use equation (3.7) to estimate $p_{\mathcal{T}}(\mathrm{y})$ or equation (3.8) to estimate class ratios directly.

Other approaches involve learning class-dependent weights to match the mean conditional features of source data with the mean marginal features of target data in a reproducing kernel Hilbert space (e.g. Iyer et al. [95], Zhang et al. [96]), or require density estimation to model $p(\mathbf{x} \mid \mathrm{y})$ (e.g. Chan and Ng [97], Storkey [98]).

3.2.2.2 Conditional shift

In the *conditional shift* setting, the marginal distribution of labels is constant and the conditional of features given labels may change across domains. This scenario is represented in Figure 3.2, from which it becomes clear that the joint distribution takes the form $p_{\mathcal{D}}(\mathbf{x}, \mathrm{y}) = p(\mathrm{y}) p_{\mathcal{D}}(\mathbf{x} \mid \mathrm{y})$, where $p(\mathrm{y})$ is domain-invariant. Besides being less realistic

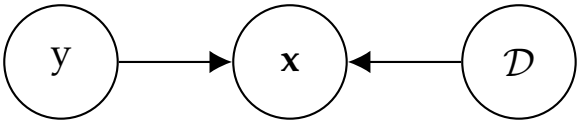


FIGURE 3.2: Graphical representation of the conditional shift setting as a Bayesian network.

than other assumptions, DA under conditional shift is in general an ill-posed problem. Nonetheless, Zhang et al. [96] show that identifiability of $p_{\mathcal{T}}(\mathbf{x} \mid \mathrm{y})$ holds when it is assumed that, for any given y, $p_{\mathcal{T}}(\mathbf{x} \mid y)$ only differs from $p_{\mathcal{S}}(\mathbf{x} \mid y)$ in location and scale and derive a kernel-based approach to estimate these parameters.

3.2.2.3 Concept shift

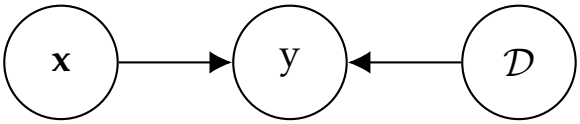


FIGURE 3.3: Graphical representation of the concept shift setting as a Bayesian network.

Concept shift refers to the situation where the marginal feature distributions $p(\mathbf{x})$ are constant but the conditional distribution of the target variable $p_{\mathcal{D}}(\mathrm{y} \mid \mathbf{x})$ is domain-dependent, thus $p_{\mathcal{D}}(\mathbf{x}, \mathrm{y}) = p(\mathbf{x})p_{\mathcal{D}}(\mathrm{y} \mid \mathbf{x})$, as implied by Figure 3.3. When the change happens over time, this setting is also known as *concept drift* (Webb et al. [99]). The literature on concept drift is vast and focuses mostly on the detection of its occurrence so that the model can be updated using new data. Gama et al. [100] overview the most popular techniques to address this problem.

3.2.2.4 Covariate shift

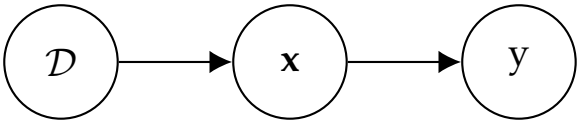


FIGURE 3.4: Graphical representation of the covariate shift setting as a Bayesian network.

Covariate shift is by far the most common assumption and therefore the most widely addressed setting in the DA literature. As implied by the graphical representation in Figure 3.4, here the conditional distribution of labels given features is constant and the marginal distribution of features is domain-dependent. Thus, $p_{\mathcal{D}}(\mathbf{x}, \mathrm{y}) = p(\mathrm{y} \mid \mathbf{x})p_{\mathcal{D}}(\mathbf{x})$, where $p(\mathrm{y} \mid \mathbf{x})$ is constant across domains. This assumption might hold for instance in

image classification problems where the domain shift is caused by different sensors or lighting conditions.

Since $p_{\mathcal{T}}(\mathrm{y} \mid \mathbf{x}) = p_{\mathcal{S}}(\mathrm{y} \mid \mathbf{x})$, infinite labeled data from the source domain and a consistent estimator of this conditional distribution would solve this DA task. However, the former is obviously unrealistic and therefore the problem should be analyzed taking into account that the available data is finite. Let $\ell(\cdot, \cdot)$ be any loss function for the supervised learning problem. The goal is then to find an unbiased estimator of the target loss $\mathbb{E}_{\mathbf{x},\mathrm{y}\sim p_{\mathcal{T}}}\left[\ell(h(\mathbf{x}), \mathrm{y})\right]$ when no labeled samples from the target domain are available. Following Sugiyama et al. [101], if the marginal densities $p_{\mathcal{S}}(\mathbf{x})$ and $p_{\mathcal{T}}(\mathbf{x})$ are known and assuming $\mathrm{Supp}(p_{\mathcal{T}}(\mathbf{x})) \subseteq \mathrm{Supp}(p_{\mathcal{S}}(\mathbf{x}))$, this estimator can be obtained using *importance weights* $w(\mathbf{x}) \triangleq p_{\mathcal{T}}(\mathbf{x})/p_{\mathcal{S}}(\mathbf{x})$:

$$\begin{aligned}
\mathbb{E}_{\mathbf{x},\mathrm{y}\sim p_{\mathcal{T}}}\left[\ell(h(\mathbf{x}), \mathrm{y})\right] &= \sum_{\mathrm{y}} \int l(h(\mathbf{x}), \mathrm{y}) p_{\mathcal{T}}(\mathbf{x}, \mathrm{y})\, \mathrm{d}\mathbf{x} \\
&= \sum_{\mathrm{y}} \int \ell(h(\mathbf{x}), \mathrm{y}) p(\mathrm{y} \mid \mathbf{x}) p_{\mathcal{T}}(\mathbf{x})\, \mathrm{d}\mathbf{x} \\
&= \sum_{\mathrm{y}} \int \frac{p_{\mathcal{T}}(\mathbf{x})}{p_{\mathcal{S}}(\mathbf{x})} \ell(h(\mathbf{x}), \mathrm{y}) p(\mathrm{y} \mid \mathbf{x}) p_{\mathcal{S}}(\mathbf{x})\, \mathrm{d}\mathbf{x} \\
&= \sum_{\mathrm{y}} \int w(\mathbf{x}) \ell(h(\mathbf{x}), \mathrm{y}) p_{\mathcal{S}}(\mathbf{x}, \mathrm{y})\, \mathrm{d}\mathbf{x} \\
&= \mathbb{E}_{\mathbf{x},\mathrm{y}\sim p_{\mathcal{S}}}\left[w(\mathbf{x}) \ell(h(\mathbf{x}), \mathrm{y})\right]. \qquad (3.9)
\end{aligned}$$

Hence, $(1/n)\sum_{i=1}^{n} w(\boldsymbol{x}_i)\ell(h(\boldsymbol{x}_i), y_i)$ is an unbiased estimator of the target loss, constructed using only source data, which can therefore be used as the loss for the supervised learning problem. When the marginal distributions are unknown and the feature space is low-dimensional, kernel density estimation techniques can be employed to estimate them (e.g. Sugiyama et al. [101], Shimodaira [102], Cortes et al. [103]). An alternative that removes the constraint on the support of the marginal target distribution is learning a transformation $T : \mathcal{X} \mapsto \mathcal{X}$ such that $p_{\mathcal{S}}(T(\mathbf{x})) = p_{\mathcal{T}}(\mathbf{x})$, which can be accomplished using optimal transport theory (Courty et al. [104]). Later approaches aim to relax the covariate shift assumption by trying to align the joint source and target distributions (Courty et al. [105]) and extend the same principles to the multi-source setting (Turrisi et al. [106]).

For high-dimensional data (e.g. images), where the marginal distributions are typically unknown and hard to estimate from finite data, deep learning models play an important role by providing a successful tool learn semantically rich low-dimensional feature representations. It is then possible to learn a function $g : \mathcal{X} \mapsto \mathcal{Z}$ mapping input data to a

new feature space $\mathcal{Z}$ such that $p_{\mathcal{T}}(g(\mathbf{x}))/p_{\mathcal{S}}(g(\mathbf{x})) \approx 1$, i.e. where the marginal feature distributions of the two domains coincide. This approach has the additional benefit of moving the overlapping support assumption to a lower-dimensional space, where it is less likely to be violated than in the original space (e.g. pixel space). In this new feature space, the covariate shift has vanished and therefore any classifier $h : \mathcal{Z} \mapsto \mathcal{Y}$ that achieves low error on the source domain will also perform well in the target. An alternative way of motivating these approaches follows from analyzing again the target risk bound provided in equation (3.4). When we presented this bound, we assumed for simplicity that the feature space was fixed and therefore the upper bound would be minimized by minimizing the source risk. However, if the feature space can itself be optimized, the term $\widehat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T})$ can be minimized by finding a feature space where the source and target marginal distributions coincide. This idea has been exploited extensively in recent years, either by matching the distributions using maximum mean discrepancy (e.g. Long et al. [84], Guo et al. [89]) or, in most cases, using an adversarial neural network (e.g. Ganin and Lempitsky [69], Zhao et al. [72], Schoenauer-Sebag et al. [91], Pei et al. [107]).

Adversarial-based DA was originally introduced by Ganin and Lempitsky [69] and results from the observation that computing $\widehat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T})$ is equivalent to finding a classifier that maximally discriminates between samples of the source and target domains (Lemma 3.1). Intuitively, if no classifier exists that can distinguish between source and target features, then the distributions of these two must coincide. Thus, if we have access to sets $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{T}}$ of labeled samples from the source domain and unlabeled samples from the target, respectively, we can train a feature extractor network $g : \mathcal{X} \mapsto \mathcal{Z}$, a classifier $h : \mathcal{Z} \mapsto \mathcal{Y}$, and a domain discriminator $d : \mathcal{Z} \mapsto \{0,1\}$ to solve the following minimax problem*:

$$\min_{g,h} \max_{d} \quad \widehat{\epsilon}_{\mathcal{S}}(h \circ g) + 1 - \left(\frac{1}{n} \sum_{\boldsymbol{x}: d(g(\boldsymbol{x}))=1} \mathbf{1}_{\boldsymbol{x} \in \widehat{\mathcal{S}}} + \frac{1}{n} \sum_{\boldsymbol{x}: d(g(\boldsymbol{x}))=0} \mathbf{1}_{\boldsymbol{x} \in \widehat{\mathcal{T}}} \right), \tag{3.10}$$

where $\circ$ denotes function composition. Several variations to this idea have been proposed so far, aiming to extend it to the multi-source setting (Zhao et al. [72]), or beyond the covariate shift assumption (Pei et al. [107]), or both (Schoenauer-Sebag et al. [91]).

*This objective is merely formal since the 0-1 loss is non-smooth and intractable. In practice, the usual classification losses are used instead.

3.2.2.5 Invariance of causal mechanisms

The settings we have discussed so far consider all features $\mathbf{x}$ as atomic and hence do not take into account how different features interact to produce the target variable y. Other approaches drop this limitation by decomposing the feature vector into individual features $x_1, x_2, \ldots, x_m$, taking into account the structure of the causal Bayesian network governing the data generating process, and using causal inference tools to identify the target distribution. These methods assume that the flow of cause and effect cannot be reversed by domain shift and therefore changes in distribution are due to different interventions in the same causal graph $\mathcal{G}$ (e.g. presence of additional exogenous variables inducing non-causal associations between features and the target variable). Bareinboim and Pearl [108] assume $\mathcal{G}$ is known and all interventions are perfect (i.e. all interventions consist of edge removal operations) and, given some subset $\bar{\mathbf{x}} \subset \mathbf{x}$, derive conditions for identifiability of $p_{\mathcal{T}}(y \mid \bar{\mathbf{x}})$ given $\mathcal{G}$ and $p_{\mathcal{S}}(\mathbf{x}, y)$. Rojas-Carulla et al. [109] and Magliacane et al. [110] relax the covariate shift setting by assuming that there exists a strict subset $\bar{\mathbf{x}} \subset \mathbf{x}$ such that $p_{\mathcal{D}}(y \mid \bar{\mathbf{x}})$ is domain-invariant and propose algorithms to infer $\bar{\mathbf{x}}$ given data from multiple source domains.

Despite being (arguably) more trustworthy than purely data-driven approaches, causality-based methods still struggle to be applied in practice. First of all, they depend to some extent on $\mathcal{G}$ being given. In many applications, domain knowledge is insufficient to build such a graph. Moreover, causal discovery algorithms, which aim to learn the structure of the graph from data, are computationally expensive and, given observational data, can only recover $\mathcal{G}$ up to its Markov equivalence class, at least when no parametric assumptions are made (Peters et al. [111]). Furthermore, these methods are unsuitable to be applied to image data as no meaningful causal reasoning can be built in pixel space and even deep neural networks are still incapable of finding suitable representations for this goal (Schölkopf et al. [112]).

3.3 Adversarial domain adaptation for object counting in videos

3.3.1 Motivation

As different sensors are added to and excluded from a network, we should take into account the fact that the sensors used at training time are different from the ones where

the model will make predictions. Since domain shifts between different sensors in a 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. [113] proposed an algorithm called Temporal Attentive Alignment for implementing DA in video datasets that explicitly attends to temporal dynamics and Liu and Li [114] proposed a spatio-temporal DA model named TrCbrBoost for classifying land use. It should be noted, though, that both only deal with a single-source-single-target setting, which is simpler than the multi-source case present when dealing with sensor networks.

For dealing with a multi-source setting, adversarial approaches have proven successful. Zhao et al. [72] introduced an algorithm called multi-source domain adversarial networks (MDAN) that makes use of k domain discriminators that aim to distinguish between the target and the k source domains. 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 consider it is worthy to investigate the adaptation of this model so that it can receive a temporal sequence as an input.

The setting where sensors correspond to video cameras is especially interesting since video data is very high-dimensional. Moreover, the fact that different cameras are located in different places, and therefore have distinct points of view, increases the domain shift and therefore makes this task even more challenging.

Here, we shall explore how to adapt the MDAN model so that both of its adversarial networks are LSTM-based. We decided to go with this type of network since it has already shown promising results in our task: Zhang et al. [115] introduced an LSTM network architecture for counting vehicles in images obtained from city cameras and reported an improvement of the mean absolute error when compared to other state of the art methods.

3.3.2 MDAN: Multi-source domain adversarial networks

We now review the MDAN model introduced by Zhao et al. [72]. This model is motivated by the target risk bound provided in Theorem 3.3 and is an extension to the multi-source setting of the single source model by Ganin and Lempitsky [69]. Specifically, the following

formal objective is considered:

$$\min_{g,h} \max_{j\in\{1,\cdots,k\}} \quad \widehat{\epsilon}_{\mathcal{S}_j}(h\circ g) + \frac{1}{2}\widehat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}_j^g, \mathcal{T}^g), \tag{3.11}$$

where $g : \mathcal{X} \mapsto \mathcal{Z}$ and $h : \mathcal{Z} \mapsto \mathcal{Y}$ are, as before, the feature extractor and the classifier networks and $\mathcal{S}_j^g$ and $\mathcal{T}^g$ are, respectively, the j-th source domain and the target domain representations in the new feature space $\mathcal{Z}$ (i.e. $p_{\mathcal{D}^g}(\mathbf{x}) \triangleq p_{\mathcal{D}}(g(\mathbf{x}))$ for any domain $\mathcal{D}$). Here, the empirical $\mathcal{H}\Delta\mathcal{H}$-divergence between $\mathcal{S}_j^g$ and $\mathcal{T}^g$ is also implemented with an adversarial domain discriminator $d_j : \mathcal{Z} \mapsto \{0,1\}$ aiming to discriminate between samples of the two domains. Thus, the model comprises k domain discriminator networks, i.e. one for each source domain. This objective is therefore identical to equation (3.10) with the only difference that, since here there are multiple source domains, the model is optimized for the hardest source domain at each training iteration. In this formulation, $\boldsymbol{\alpha}$ in equation (3.5) is a one-hot vector whose active component corresponds to the hardest source domain. Because the bound holds for any convex combination of source domains, the authors also explore a soft-max version of this problem, where smaller positive weights are assigned to the easier source domains:

$$\min_{g,h} \quad \frac{1}{\gamma}\log\sum_{j=1}^{k}\exp\left(\gamma(\widehat{\epsilon}_{\mathcal{S}_j}(h\circ g) + \frac{1}{2}\widehat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}_j^g, \mathcal{T}^g))\right). \tag{3.12}$$

Here, $\gamma > 0$ is a hyperparameter controlling the softness of the max operation ($\gamma \to \infty$ corresponds to the hard-max). In our experiments, we will also evaluate the scenario where the weights $\boldsymbol{\alpha}$ are equal for all domains, i.e. where the multi-domain loss consists of the simple average of the losses across source domains.

3.3.2.1 The gradient reversal layer

Adversarial DA algorithms, of which MDAN is a particular case, all aim to solve some variant of the following minimax problem:

$$\min_{\boldsymbol{\theta}_g,\boldsymbol{\theta}_h} \max_{\boldsymbol{\theta}_d} \quad \left\{L(\boldsymbol{\theta}_g,\boldsymbol{\theta}_h,\boldsymbol{\theta}_d) = L_{\text{task}}(\boldsymbol{\theta}_g,\boldsymbol{\theta}_h) - \mu_d L_{\text{disc}}(\boldsymbol{\theta}_g,\boldsymbol{\theta}_d)\right\}, \tag{3.13}$$

where L_{task} is the supervised loss for the desired task, L_{disc} is the classification loss for the domain discrimination task, $\boldsymbol{\theta}_g$, $\boldsymbol{\theta}_h$, and $\boldsymbol{\theta}_d$ are the parameters of the feature extractor, task classifier, and domain discriminator networks, respectively, and $\mu_d > 0$ is a hyperparameter. Larger values of μ_d imply a stronger enforcement of domain-invariant representations. This is a treatable surrogate of objective (3.10).

Solving this problem then consists in finding a saddle point of this loss function. Using automatic differentiation libraries (e.g. PyTorch or TensorFlow), the naive solution would involve declaring two optimizers, one for parameters $\boldsymbol{\theta}_g$ and $\boldsymbol{\theta}_h$ and another for $\boldsymbol{\theta}_d$, and then performing gradient descent with the former and gradient ascent with the latter:

$$\boldsymbol{\theta}_g \leftarrow \boldsymbol{\theta}_g - \rho \left(\nabla_{\boldsymbol{\theta}_g} L_{\text{task}} - \mu_d \nabla_{\boldsymbol{\theta}_g} L_{\text{disc}} \right), \tag{3.14}$$

$$\boldsymbol{\theta}_h \leftarrow \boldsymbol{\theta}_h - \rho \nabla_{\boldsymbol{\theta}_h} L_{\text{task}}, \tag{3.15}$$

$$\boldsymbol{\theta}_d \leftarrow \boldsymbol{\theta}_d + \rho \left(-\mu_d \nabla_{\boldsymbol{\theta}_d} L_{\text{disc}} \right), \tag{3.16}$$

where $\rho > 0$ is the learning rate. This implies an extra computational burden because gradients need to be backpropagated through the discriminator network twice, one for computing $\nabla_{\boldsymbol{\theta}_d} L_{\text{disc}}$ and another for computing $\nabla_{\boldsymbol{\theta}_g} L_{\text{disc}}$.

The gradient reversal layer (Ganin and Lempitsky [69]) is an ingenious solution to this problem. This layer is a pseudo-function r that behaves as the identify in the forward pass but inverts the sign of the gradient in the backward, i.e.:

$$r(\boldsymbol{x}) \triangleq \boldsymbol{x}, \quad \frac{\partial r}{\partial \boldsymbol{x}} \triangleq -I. \tag{3.17}$$

By placing it in between the feature extractor g and the domain discriminator d, the gradient $\nabla_{\boldsymbol{\theta}_g} L_{\text{disc}}$ will come with its sign inverted. Thus, performing gradient descent over all parameters of

$$L_{\text{task}}(\boldsymbol{\theta}_g, \boldsymbol{\theta}_h) + \mu_d L_{\text{disc}}(\boldsymbol{\theta}_g, \boldsymbol{\theta}_d), \tag{3.18}$$

yields exactly update equations (3.14), (3.15), and (3.16).

3.3.3 FCN-rLSTM: Spatio-temporal deep neural network for object counting

Zhang et al. [115] proposed FCN-rLSTM, a deep neural network architecture for counting vehicles in low-quality videos captured by city cameras that will constitute the backbone of our models.

Although each video frame should contain all the information required to identify the number of vehicles in it, issues with the quality of the collected data can make vehicle counting a difficult problem, namely low resolution, vehicle occlusion, and different vehicle scales, particularly noticeable when the camera is too close to the road. Thus, assuming that the frame rate is sufficiently large when compared to the vehicles speed, leveraging

information from the previous frames should help improving the accuracy of the prediction for the current frame. For this reason, FCN-rLSTM combines a fully convolutional network with a recurrent module, which preserves memory from the previous frames in LSTM cells. This is a density-based estimation method, being able to deal well with low frame rates, low resolutions, and vehicle occlusions, but having difficulty in accounting for different vehicle scales.

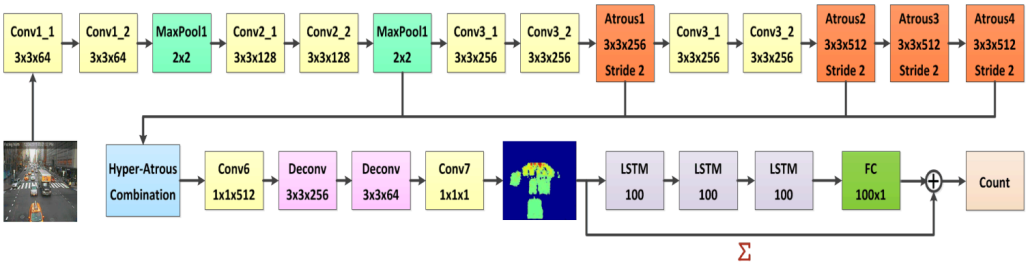


FIGURE 3.5: Architecture of the FCN-rLSTM model (reprinted from Zhang et al. [115]).

Figure 3.5 shows the full architecture of this model, from which it can be observed that the convolutional part outputs a density map $\widehat{\boldsymbol{D}}$. The density maps are normalized so that the sum of the pixels corresponding to each vehicle sum up to 1 and, therefore, the sum of all pixels in the density map sup to the total number of vehicles in the frame. Thus, the final predicted count $\widehat{y}^{(t)}$ is the result of this sum plus a residual provided by a recurrent neural network, which aims to correct the predicted count by leveraging information from the previous frames.

Training this model implies that each input frame $\boldsymbol{X}^{(t)}$ is annotated with the corresponding ground-truth density map $\boldsymbol{D}^{(t)}$ and vehicle count $y^{(t)}$. The loss function is defined as:

$$L(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}||\widehat{\boldsymbol{D}}_i^{(t)} - \boldsymbol{D}_i^{(t)}||^2 + \frac{\lambda_c}{n}\sum_{i=1}^{n}||\widehat{y}_i^{(t)} - y_i^{(t)}||^2, \tag{3.19}$$

where $\lambda_c > 0$ is a hyperparameter controlling the relative weight of the vehicle counting loss, and the dependency of $\widehat{\boldsymbol{D}}_i^{(t)}$ and $\widehat{y}_i^{(t)}$ on the network parameters $\boldsymbol{\theta}$ is omitted to ease the notation.

3.3.4 Combining MDAN and FCN-rLSTM

We now explore several possibilities to combine MDAN and FCN-rLSTM into a single model capable of accurately counting vehicles in images from a target camera, provided that at training time we only have access to annotated data from other cameras and unlabeled data from the target.

3.3.4.1 Non-Temporal model

The non-temporal model uses a sub-network in the FCN-rLSTM model as its backbone. It consists in a simplification of this, by not using LSTMs or making any other consideration on the temporal or sequential nature of the data. Figure 3.6 shows the architecture of the non-temporal model.

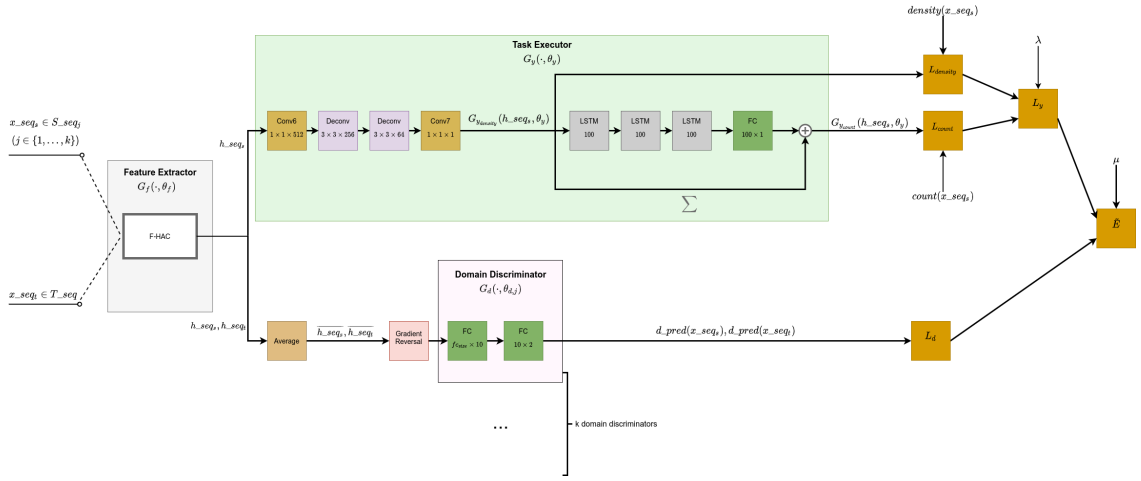


FIGURE 3.6: Non-Temporal model. Reprinted from de Andrade [61].

The component F-HAC in that figure corresponds to all layers of the FCN-rLSTM up to the hyper-atrous combination. It is used as a feature extractor, whereas the rest of the convolutional layers are used for the object counting task. The k domain discriminators consist of two fully connected layers which are preceded by a gradient reversal layer.

This model will be our baseline as we are primarily interested in merging the benefits of DA techniques and sequential modeling.

3.3.4.2 SingleLSTM model

The SingleLSTM model uses the whole FCN-rLSTM model for the desired regression task, but the domain discriminators are still non-sequential, as shown in Figure 3.7. Thus, in this model, the domain discrimination task only takes into account the domain-specific information provided by each frame individually and does not account for the domain-specific temporal dynamics that may exist.

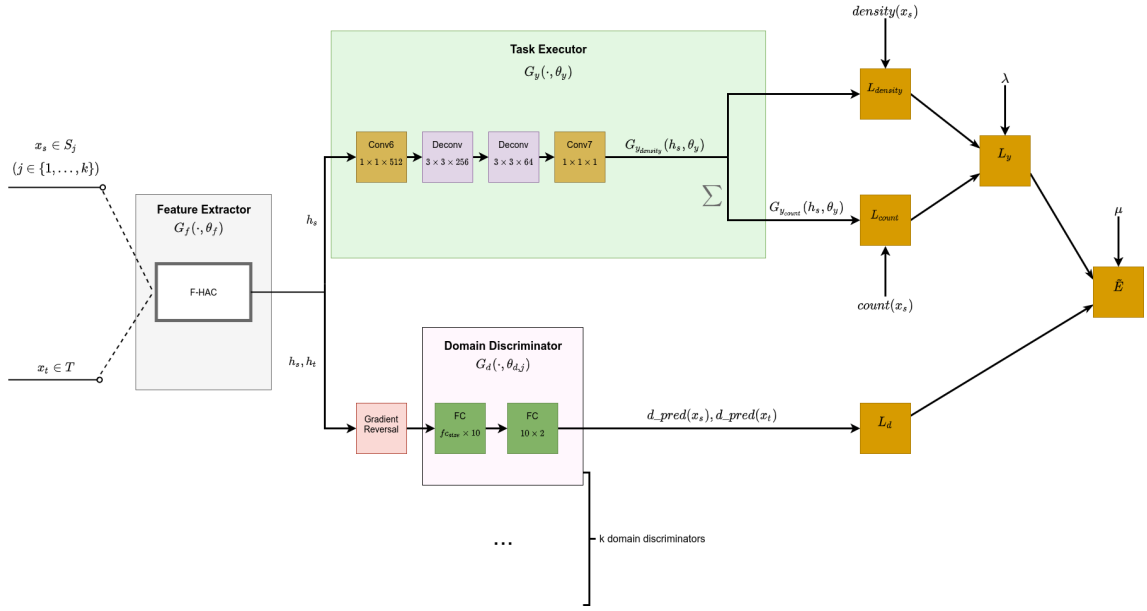


FIGURE 3.7: Temporal regression model. Reprinted from de Andrade [61].

3.3.4.3 DoubleLSTM model

The DoubleLSTM model overcomes the limitations of the SingleLSTM model by incorporating three LSTM layers in each domain discriminator. These aim to learn the domain-specific temporal dynamics that might be helpful for the discrimination task. The schematic is in Figure 3.8.

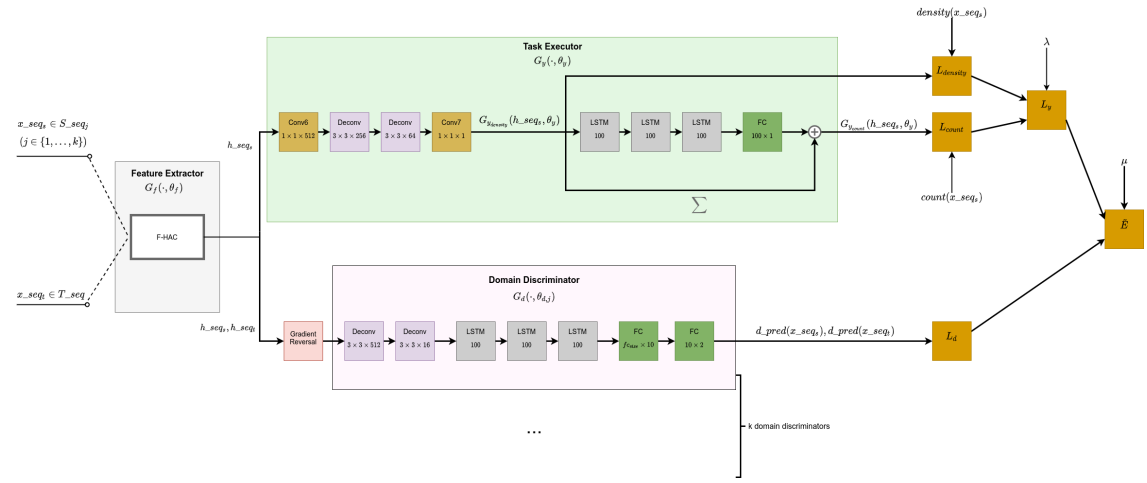


FIGURE 3.8: Double-Temporal model. Reprinted from de Andrade [61].

3.3.4.4 CommonLSTM model

An alternative to the DoubleLSTM model is to extract temporal features that are common to the desired object counting task and to the domain discrimination task. This is accomplished by the CommonLSTM model, which pushes the domain discriminators after the LSTMs and just before the final fully connected layer, as shown in Figure 3.9.

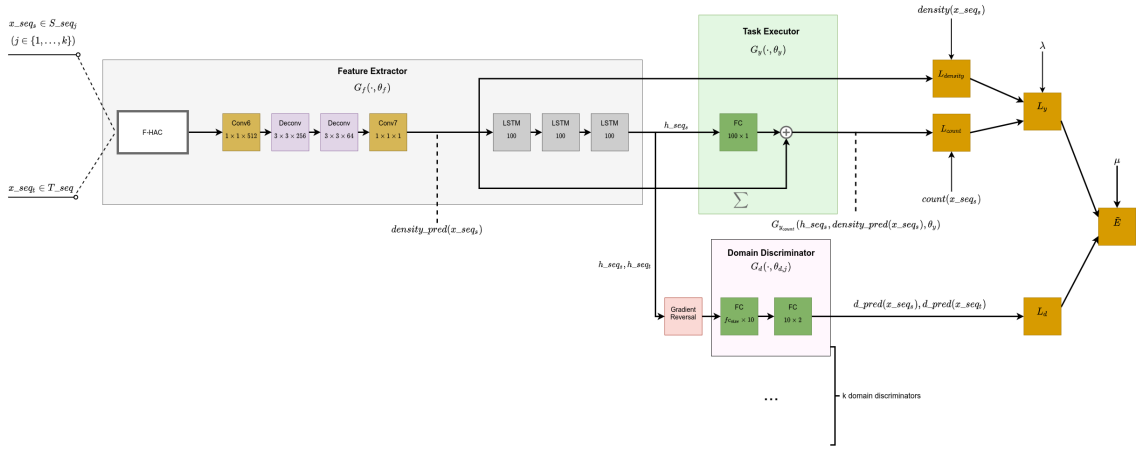


FIGURE 3.9: Common-Temporal model. Reprinted from de Andrade [61].

This model is fundamentally different from all previous ones since here the feature space where domain-invariance is promoted happens much later in the network. Specifically, the density maps $\widehat{D}$ are a few layers earlier, so they should not be very much affected by the domain-invariance constraint. This might have a beneficial effect on the performance since density maps are defined by the positions in the image where vehicles appear. Since these positions depend on the shape of the street that each camera is capturing, domain-invariant density maps will certainly be inaccurate.

3.3.4.5 Overview

All the proposed models have specific strengths and weaknesses that would make them suitable for certain scenarios and unlikely to show a good performance in others. Here, we anticipate a few of those scenarios.

If the video frame rate is low compared to the objects speed, the number of objects in a given frame is not a good predictor for the number of objects in a subsequent frame. In this setting, the non-temporal model is ideal, as it does not make any temporal consideration. In this case, using a temporal model would have a likely harmful effect on the model performance.

If the frame rate is sufficiently high and the temporal dynamics in the target domain are close to the dynamics in the sources, the SingleLSTM model is likely the best choice. By using an LSTM for the task executor but none for the domain discriminators, it will be able to make small adjustments in the predicted object count and avoid the unnecessary computational cost introduced by having a sequential model in the domain discriminator.

When there is a strong correlation in the object count in consecutive video frames and the dynamics in the target domain are dissimilar to the sources, it may be beneficial to employ sequential models in both the task executor and domain discriminators. Thus, DoubleLSTM and CommonLSTM models are likely to outperform the remaining. Their approach differs, as DoubleLSTM model uses an LSTM network for the task executor and another one for each domain discriminator, whereas CommonLSTM includes an LSTM network in the feature extractor. As the common-temporal model will not enforce similarity between the predicted density maps as much as the double-temporal version, it is probably the best choice when the density maps of the target domain differ significantly from the sources.

3.3.5 Experiments

3.3.5.1 Experimental protocol

We will run experiments with the domain adaptation models presented in Section 3.3.4. Additionally, we will use models FCN-HA and FCN-rLSTM as baselines, which do not employ any technique to tackle domain shift. The former is a subnetwork of the latter consisting of all layers excluding the LSTM network, i.e. it coincides with the Non-Temporal model when its domain discriminator is removed. Therefore, it does not account for any temporal correlations between consecutive frames.

In every experiment, assume we have $k+1$ domains $\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_{k+1}$. In a domain adaptation scenario, we do $k+1$ runs, so that in run number t, domain $\mathcal{D}_t$ will be chosen as the target domain and all the others will be source domains. For each experiment, we have computed results in both unsupervised and semi-supervised settings, described 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 $\mathcal{D}_t$.

- Semi-supervised: For semi-supervised results, we have a validation dataset that corresponds to 30% of the samples extracted from target $\mathcal{D}_t$. We use those samples to select the best model obtained throughout the training epochs. This model is then tested with the remaining 70% of the samples from $\mathcal{D}_t$.

For each dataset, we ran a total of 5 experiments for the DA models, where we varied the value of hyperparameter μ_d in the range $[10^{-5}, 10^{-1}]$. Recall that μ_d controls the weight to give to the domain discrimination loss, relative to the task execution loss.

In all experiments, we train the model for 50 epochs, using Adam optimizer (Kingma and Ba [47]) with a learning rate of 10^{-4}. The adopted evaluation metric is the Mean Absolute Error (MAE) between the predicted object count and the ground truth for each frame.

3.3.5.2 UCSDPeds dataset

UCSDPeds (Chan and Vasconcelos [116]) 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 $\times$ 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 coordinates of the central point of every person in the frame.

Each camera point of view corresponds to a domain in our experiments, named *vidd* and *vidf*. Figures 3.10 and ´3.11 show examples of frames in each one of the two domains.

FIGURE 3.10: Domain *vidd* from the UCSDPeds dataset.

FIGURE 3.11: Domain *vidf* from the UCSDPeds dataset.

Table 3.1 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.

	mean	std. dev.
vidd	6.448	3.089
vidf	27.570	7.393

TABLE 3.1: Mean number of people and respective standard deviation for each domain in the UCSDPeds dataset.

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 3.12 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 mirrored image of each frame to the dataset.

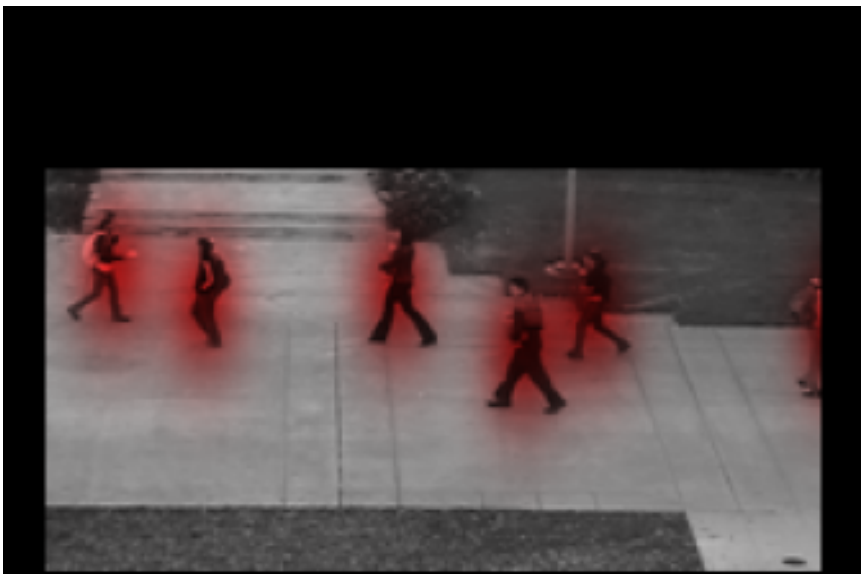


FIGURE 3.12: Sample density map for the UCSDPeds dataset.

3.3.5.3 WebCamT dataset

WebCamT (Zhang et al. [117]) 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 14 cameras that cover multiple scenes, camera perspectives, congestion states, and weather conditions. We chose four cameras from those 14 to correspond to our domains. Each of those four 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 3.13-3.16 show examples of frames in each of the four domains. Table 3.2 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 number of vehicles in each frame differs significantly across domains, being more than three times larger in domain *691* than in domain *846*. Hence, there is significant target shift in this dataset and, consequently, the covariate shift assumption does not hold.

FIGURE 3.13: Domain *511* from WebCamT dataset.

FIGURE 3.14: Domain *551* from the WebCamT dataset.

FIGURE 3.15: Domain *691* from the WebCamT dataset.

FIGURE 3.16: Domain *846* from the WebCamT dataset.

	mean	std. dev.
511	12.039	6.32
551	18.362	4.21
691	25.470	10.738
846	6.873	2.477

TABLE 3.2: Mean number of vehicles and respective standard deviation for each domain in the WebCamT dataset.

After extracting each frame from the videos, we computed its density map by placing a 4×4 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 3.17 shows an example of a density map in a resized frame, where the Gaussian kernel around each car is shown in red.

FIGURE 3.17: Sample density map for the WebCamT dataset.

3.3.6 Choice of optimization problem

The DA task for the UCSDPeds dataset is single source, so there is no choice to be made regarding the combination of source domains. In this setting, the MDAN model boils down to the original adversarial DA algorithm proposed by Ganin and Lempitsky [69], where one single domain discriminator is used to distinguish between source and target samples.

In the WebCamT dataset, we have three different source domains, so this is a multi-source scenario. As discussed in Section 3.3.2, we consider three possible optimization problems for our DA models to solve, namely hard-max, soft-max, and average. Table 3.3 shows results for each of our models on these formulations, evaluated in an unsupervised setting. These suggest that averaging tends to perform better than the other two approaches in this dataset. This is somewhat intuitive: 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 use any of the other two optimization problems, our models would dedicate an outweighed importance

to the hardest source domain, disregarding the remaining. For this reason, we shall adopt the averaging formulation in all subsequent experiments.

	Hard-max	Soft-max	Average
Non-Temporal	18.15	11.16	**9.91**
SingleLSTM	10.54	**9.33**	11.10
DoubleLSTM	9.59	7.73	**6.14**
CommonLSTM	**9.85**	12.07	10.22

TABLE 3.3: Comparison of different optimization problems. The values indicate the average MAE count across domains in the WebCamT dataset. The experiments were run with $\mu_d = 10^{-3}$.

3.3.7 Unsupervised setting

Table 3.4 shows the results in the unsupervised setting for both datasets. For each domain, the best MAE obtained with different values of μ_d is shown.

	UCSDPeds			WebCamT				
	vidd	*vidf*	Avg	*511*	*551*	*691*	*846*	Avg
FCN-HA	15.23	7.14	11.19	**5.01**	**2.58**	17.22	4.67	7.37
FCN-rLSTM	11.04	5.98	8.51	6.26	8.79	13.63	6.31	8.75
Non-Temporal	**6.24**	22.52	14.38	9.63	3.50	18.27	5.58	9.85
SingleLSTM	7.75	9.07	9.32	6.58	5.10	15.26	9.55	10.53
DoubleLSTM	8.04	**3.89**	6.49	5.01	4.58	10.97	**3.03**	**6.15**
CommonLSTM	6.83	4.80	**6.18**	6.11	5.55	**10.51**	3.56	8.15

TABLE 3.4: MAE count per domain in UCSDPeds and WebCamT datasets (unsupervised setting). Column Avg indicates the average MAE count across domains.

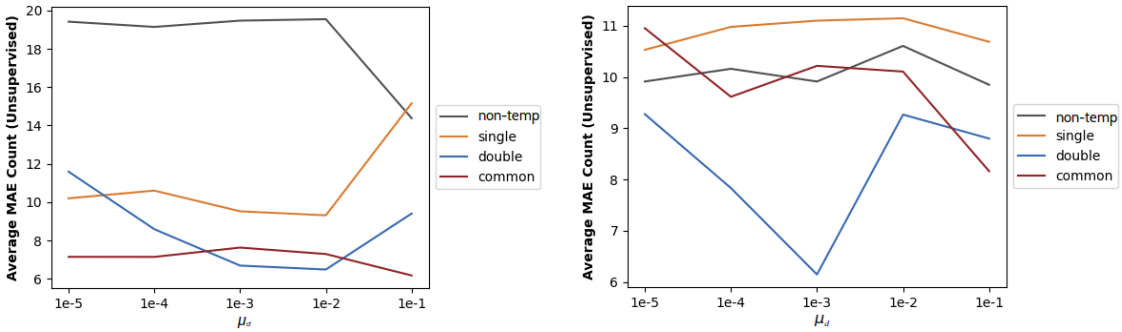
In UCSDPeds, CommonLSTM showed the best results, followed by DoubleLSTM. In fact, any LSTM-based model showed a better average performance than any model where LSTMs are absent. This validates our initial hypothesis that making temporal considerations would significantly improve the accuracy. When it 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 average MAE count. Still, in this case, the advantage of promoting domain-invariant representations is not as clear as the advantage of leveraging temporal information. For example, the Non-Temporal model, that includes a domain discriminator, showed worse results than FCN-rLSTM.

In WebCamT, even though the double-temporal 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 average MAE count 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.

We find that the model with the most balanced performance across domains and datasets was the DoubleLSTM as its MAE count in a given domain was always in the top-3. This observation coupled with the fact that the DoubleLSTM was also the model with the best average MAE count in UCSDPeds and the second best in WebCamT allows us to conclude that this method was the best-performing overall.

Results regarding the sensitivity to the hyperparameter μ_d are shown in Figure 3.18.



(A) UCSDPeds (B) WebCamT

FIGURE 3.18: Average MAE count across domains for each μ_d in the unsupervised setting in UCSDPeds and WebCamT datasets.

This plot confirms that the DoubleLSTM model was the best-performing in the WebCamT dataset and is at least comparable to CommonLSTM in UCSDPeds. Setting $\mu_d = 10^{-3}$ achieves close to optimal performance for all models in both datasets, except for Non-Temporal in UCSDPeds and CommonLSTM in WebCamT, whose performance improves significantly for larger values of μ_d.

3.3.8 Semi-supervised setting

Table 3.5 shows the results in the semi-supervised setting for both datasets. Like before, the best MAE obtained with different values of μ_d is shown.

There is a noticeable difference between these results and those for the unsupervised setting (Table 3.4), with the former being significantly better that the latter. As the experiments we ran with the WebCamT had more data and more domains, the resulting models have smaller variance, which caused the difference in accuracy between the two settings not to be as wide as in UCSDPeds.

	UCSDPeds			WebCamT				
	vidd	*vidf*	Avg	*511*	*551*	*691*	*846*	Avg
FCN-HA	1.43	4.48	2.96	**2.05**	**2.59**	6.56	2.394	**3.40**
FCN-rLSTM	2.01	**2.07**	2.04	5.40	3.49	9.15	3.54	5.39
Non-Temporal	0.84	3.49	2.21	3.11	3.57	10.96	2.05	5.04
SingleLSTM	0.94	5.07	3.00	4.24	4.39	7.03	**1.52**	4.34
DoubleLSTM	0.95	2.84	**1.89**	4.22	4.55	**6.46**	1.58	4.27
CommonLSTM	**0.88**	2.61	1.94	4.23	4.54	8.67	1.63	4.98

TABLE 3.5: MAE count per domain in UCSDPeds and WebCamT datasets (semi-supervised setting). Column Avg indicates the average MAE count across domains.

In UCSDPeds, we can discern the same general remarks we made when we analyzed the unsupervised problem. 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 average MAE count. DA models proved, in general, to be better than non-DA models.

For WebCamT, again 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 average MAE count.

Please refer to de Andrade [61] for an extended discussion and further experimental results.

3.3.9 Conclusion

We have proposed several sensible model architectures to solve the problem of domain adaptation for the task of counting objects in videos. All of those result from the combination of two state of the art models, one for the object counting task (FCN-rLSTM) and the other one for multi-source DA (MDAN).

Among the proposed models, it stands out that DoubleLSTM and CommonLSTM showed a significant improvement over the non-DA model FCN-rLSTM in most cases. Models Non-Temporal and SingleLSTM, on the other hand, had a very unsatisfactory performance, by not showing better results than methods FCN-rLSTM and FCN-HA. 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 and choice of the hyperparameter μ_d. Unfortunately, informed choices about these become especially difficult in the unsupervised setting, where it is impossible to obtain a direct evaluation of how a given choice will perform on the target domain.

It should also be noted 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.

All the previous considerations prove the difficulty of the DA problem, especially when the domain shift is large as was the case in the two datasets we have used in these experiments. This discussion makes a good starting point for the next section, where a novel algorithm for unsupervised multi-source DA is presented.

3.4 Tackling multi-source domain adaptation with optimism and consistency

3.4.1 Introduction

The ability of deep neural networks to learn rich feature representations and the recent surge of adversarial learning techniques have led to numerous approaches that resort to an adversarial objective to learn those representations. The adversary is usually implemented with a domain discriminator network (Ganin and Lempitsky [69]), which aims to discriminate samples between source and target domains, and is jointly trained with the feature extractor and task classifier through a minimax game. Other models resort to the minimization of distribution dissimilarity metrics between the target and source feature distributions (Guo et al. [89], Ferreira et al. [118]). It is known, however, that if the learned features violate the covariate shift assumption, domain-invariant features shall deteriorate the generalization performance of the model on the target domain (Zhao et al. [79]) – a problem that we refer to as *the curse of domain-invariant representations*. Addressing this issue in the unsupervised setting is challenging, although some strategies have been proposed. Pei et al. [107] use a domain discriminator per class and the probability output of the task classifier as attention weights for the discriminator. Schoenauer-Sebag et al. [91] introduce a loss term that repulses unlabeled examples from the labeled ones

whenever the classifier has low confidence on classifying the unlabeled examples. Thus, both methods depend on the classifier to make correct predictions on the target samples early in the training. Here, we avoid this issue by employing a consistency loss, together with a minimum confidence threshold, that enforces agreement on the class predictions for original and augmented target samples.

Our contributions in this section are summarized as follows: i) we present a corollary of the theoretical results from Ben-David et al. [73] which motivates our methodology (Section 3.4.2); ii) we present our multi-source adversarial model which learns the distribution weights for each source domain jointly with all remaining parameters and following a theoretically-grounded mildly optimistic approach (Section 3.4.3.1); iii) we discuss how a simple consistency regularization technique may help avoid the curse of domain-invariant representations (Section 3.4.3.2); iv) we conduct extensive experiments on benchmark datasets that confirm the effectiveness of the proposed methodologies (Section 3.4.4).

3.4.2 Motivation

3.4.2.1 An upper bound on the target risk

Intuitively, if the source and target domains are similar and the amount of labeled source data is sufficiently large, a classifier trained to reach low empirical error on the source domain will likely achieve a low error on the target domain too. When multiple source domains are available, a combination of their respective data yields the optimal strategy. The following bound, which is a corollary of the results from Ben-David et al. [73], formalizes these ideas and enlightens some properties that will be exploited by our approach. For completeness, the proof is provided in Appendix B.1. In the following, Δ denotes the k-simplex and $\mathcal{S}_{\boldsymbol{\alpha}}$ is the $\boldsymbol{\alpha}$-weighted mixture of source domains (i.e. $\Delta \triangleq \{\boldsymbol{\alpha} \in [0,1]^k : \sum_{j=1}^k \alpha_j = 1$ and $p_{\mathcal{S}_{\boldsymbol{\alpha}}}(\mathbf{x}) \triangleq \sum_{j=1}^k \alpha_j p_{\mathcal{S}_j}(\mathbf{x})\}$.

Theorem 3.4. *Let $\mathcal{H}$ be a hypothesis class with VC-dimension d. Consider an unlabeled set of n samples drawn from the target domain $\mathcal{T}$ and, for each $j \in \{1, 2, ..., k\}$, a labeled set of n/k samples drawn from the source domain $\mathcal{S}_j$. Then, for any $h \in \mathcal{H}$ and any $\alpha \in \Delta$, with probability at least $1 - \delta$ over the choice of samples,*

$$\epsilon_{\mathcal{T}}(h) \leq \sum_{j=1}^{k} \alpha_j \widehat{\epsilon}_{\mathcal{S}_j}(h) + \frac{1}{2}\widehat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}_{\boldsymbol{\alpha}}, \mathcal{T}) + \lambda_{\boldsymbol{\alpha}} + B_{\boldsymbol{\alpha}}(\delta) + V(\delta), \tag{3.20}$$

where

$$B_{\boldsymbol{\alpha}}(\delta) \triangleq 2\sqrt{\frac{k}{n}\left(2d\log(2(n+1)) + \log\frac{8}{\delta}\right)\sum_{j=1}^{k}\alpha_j^2}, \tag{3.21}$$

$$\lambda_{\boldsymbol{\alpha}} \triangleq \min_{h\in\mathcal{H}} \sum_{j=1}^{k} \alpha_j \epsilon_{\mathcal{S}_j}(h) + \epsilon_{\mathcal{T}}(h), \tag{3.22}$$

$$V(\delta) \triangleq 2\sqrt{\frac{1}{n}\left(2d\log(2n) + \log\frac{4}{\delta}\right)}. \tag{3.23}$$

This bound is structurally very similar to the one presented in Theorem 3.3 (Zhao et al. [72]), with two slight differences: i) we work with the (empirical) $\mathcal{H}\Delta\mathcal{H}$-divergence between the target and the mixture of source domains directly, instead of upper-bounding it with the convex combination of (empirical) $\mathcal{H}\Delta\mathcal{H}$-divergences between the target and each source domain; and ii) we show the dependency of the bound on the quantity $B_{\boldsymbol{\alpha}}(\delta)$, whose interpretation is given in the following discussion.

3.4.2.2 The curse of domain-invariant representations

If the optimal $\boldsymbol{\alpha}$ was known, a learning algorithm for DA could, in principle, be trained to minimize the first two terms in the bound. For this purpose, it should find a function $g : \mathcal{X} \mapsto \mathcal{Z}$ in a set of feature transformations $\mathcal{F}$ and a classifier $h : \mathcal{Z} \mapsto \mathcal{Y}$ in $\mathcal{H}$. The first term in the bound is minimized by training the classifier h on the desired task, using the labeled data from all source domains. The second term is minimized by finding a feature transformation g such that the induced distributions $\mathcal{T}^g$ and $\mathcal{S}^g_{\boldsymbol{\alpha}}$ are similar. This is the main idea exploited by adversarial-based DA algorithms, which use an adversarial objective to match source and target distributions in a latent feature space. The problem with this approach is that it completely overlooks the role of the third term, $\lambda_{\boldsymbol{\alpha}}$, which corresponds to the minimum possible combined risk of a classifier in $\mathcal{H}$ on the source and target domains. Zhao et al. [79] show constructively that a low error on the source domain and domain-invariant features are insufficient to ensure low target risk and may actually have the opposite effect, by increasing $\lambda_{\boldsymbol{\alpha}}$. Sufficiency is established only under the covariate shift assumption, where the conditional distributions of labels $\mathrm{y} \in \mathcal{Y}$ given features $\mathbf{z} \in \mathcal{Z}$ are the same across source and target domains and only the marginal distributions of features differ. Since, in the unsupervised DA setting, target labels are not available, imposing or verifying covariate shift is not possible. Moreover, whenever

the marginal distributions of labels differ (target shift), a strong enforcement of domain-invariant feature representations necessarily hurts the covariate shift assumption, by imposing the marginals on features to coincide. For this reason, training adversarial-based DA algorithms for many iterations generally yields worse performance (Zhao et al. [79]). In this work, we show empirically that learning a more robust feature transformation will generally help mitigate this issue.

3.4.2.3 Choosing the combination of source domains

Looking at the first two terms of the learning bound in Theorem 3.4 suggests that $\boldsymbol{\alpha}$ could be chosen in an optimistic way, by selecting the source domain in which the sum of the risk of the classifier and the $\mathcal{H}\Delta\mathcal{H}$-divergence with respect to the target reaches the lowest value. However, the term $B_{\boldsymbol{\alpha}}(\delta)$ is proportional to the L^2-norm of $\boldsymbol{\alpha}$, which is maximum when $\boldsymbol{\alpha}$ is one-hot and minimum when $\alpha_j = 1/k, \forall\, j$. This has an intuitive explanation: choosing a sparse $\boldsymbol{\alpha}$ implies discarding data from the source domains whose component is zero, so the classifier is trained with intrinsically less data and its error tends to increase. This discussion naturally leads to the following formal objective:

$$\min_{\boldsymbol{\alpha}\in\Delta,\, h\in\mathcal{H},\, g\in\mathcal{F}} \sum_{j=1}^{k} \alpha_j \widehat{\epsilon}_{\mathcal{S}_j}(h\circ g) + \frac{1}{2}\widehat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}^g_{\boldsymbol{\alpha}}, \mathcal{T}^g) + \mu||\boldsymbol{\alpha}||, \tag{3.24}$$

where $\mu = \mu(\delta) > 0$. This objective contrasts with the idea from Zhao et al. [72], where the authors essentially choose to minimize the loss for the hardest source domain at each iteration of the learning algorithm, which is a much more pessimistic approach than ours. In either case, though, the minimum combined risk $\lambda_{\boldsymbol{\alpha}}$ is uncontrolled and, as discussed in Section 3.4.2.2, this is an issue that must be addressed.

3.4.3 Methodology

Given the discussion in the previous section, we now explain our method in detail. It basically consists of two major approaches, explained throughout this section.

3.4.3.1 Domain adaptation from a dynamic mixture of sources

We first address the problem of casting the objective (3.24) into a computationally treatable surrogate. As in many recent works in DA, we resort to neural networks to implement g and h and the empirical $\mathcal{H}\Delta\mathcal{H}$-divergence is implemented with a domain discriminator

network, which aims to distinguish between samples of $\mathcal{S}_\alpha^g$ and $\mathcal{T}^g$.* Taking the previous considerations and replacing the intractable empirical risks by the usual classification losses, objective (3.24) is cast as:

$$\min_{\boldsymbol{\alpha}\in\Delta,\boldsymbol{\theta}_g,\boldsymbol{\theta}_h} \max_{\theta_d} \left\{ \mathcal{L}(\boldsymbol{\alpha},\boldsymbol{\theta}_g,\boldsymbol{\theta}_h,\boldsymbol{\theta}_d) = \mathcal{L}_{\text{class}}(\boldsymbol{\alpha},\boldsymbol{\theta}_g,\boldsymbol{\theta}_h) - \mu_d \mathcal{L}_{\text{disc}}(\boldsymbol{\alpha},\boldsymbol{\theta}_g,\boldsymbol{\theta}_d) + \mu_s ||\boldsymbol{\alpha}||^2 \right\}, \quad (3.25)$$

where

$$\mathcal{L}_{\text{class}}(\alpha,\boldsymbol{\theta}_g,\boldsymbol{\theta}_h) = -\sum_{j=1}^{M} \alpha_j \mathbb{E}_{\mathbf{z}\sim\mathcal{S}_j^g} \left[\log p(\mathrm{y} = f_{\mathcal{S}_j^g}(\mathbf{z}) \mid \mathbf{z},\boldsymbol{\theta}_h) \right], \quad (3.26)$$

$$\mathcal{L}_{\text{disc}}(\alpha,\boldsymbol{\theta}_g,\boldsymbol{\theta}_d) = -\mathbb{E}_{\mathbf{z}\sim\mathcal{S}_\alpha^g} \left[\log p(\mathrm{d} = 0 \mid \mathbf{z},\theta_d) \right] - \mathbb{E}_{\mathbf{z}\sim\mathcal{T}^g} \left[\log p(\mathrm{d} = 1 \mid \mathbf{z},\boldsymbol{\theta}_d) \right]. \quad (3.27)$$

Here, $\boldsymbol{\theta}_h$ and $\boldsymbol{\theta}_d$ are the parameters of the classifier and domain discriminator networks, respectively, $\mu_d, \mu_s > 0$ are hyperparameters, $\mathrm{y} \in \mathcal{Y}$ is the categorical r.v. associated with the class label, $f_{\mathcal{S}_j^g} : \mathcal{Z} \mapsto \mathcal{Y}$ is the true (multiclass) labeling function for the j-th source domain, d is a Bernoulli random variable that discriminates source and target domains and the function $g = g(\cdot,\boldsymbol{\theta}_g) : \mathcal{X} \mapsto \mathcal{Z}$ is a neural network, parametrized by $\boldsymbol{\theta}_g$, mapping input samples to features. By linearity of expectations and using the fact that $\mathbf{z} = g(\mathbf{x},\boldsymbol{\theta}_g)$, for input samples $\mathbf{x} \in \mathcal{X}$, we have:

$$\mathcal{L}_{\text{class}}(\boldsymbol{\alpha},\boldsymbol{\theta}_g,\boldsymbol{\theta}_h) \triangleq -\sum_{j=1}^{k} \alpha_j \mathbb{E}_{\mathbf{x}\sim\mathcal{S}_j} \left[\log p(\mathrm{y} = f_{\mathcal{S}_j}(\mathbf{x}) \mid \mathbf{x};\boldsymbol{\theta}_g,\boldsymbol{\theta}_h) \right], \quad (3.28)$$

$$\mathcal{L}_{\text{disc}}(\boldsymbol{\alpha},\boldsymbol{\theta}_g,\boldsymbol{\theta}_d) \triangleq -\sum_{j=1}^{k} \alpha_j \mathbb{E}_{\mathbf{x}\sim\mathcal{S}_j} \left[\log p(\mathrm{d} = 0 \mid \mathbf{x};\boldsymbol{\theta}_g,\boldsymbol{\theta}_d) \right] - \mathbb{E}_{\mathbf{x}\sim\mathcal{T}} \left[\log p(\mathrm{d} = 1 \mid \mathbf{x};\boldsymbol{\theta}_g,\boldsymbol{\theta}_d) \right]. \quad (3.29)$$

In order to satisfy the constraint $\boldsymbol{\alpha} \in \Delta$, we reparametrize the model using $\alpha = \text{softmax}(\boldsymbol{\beta})$, for an unconstrained parameter $\boldsymbol{\beta} \in \mathbb{R}^k$. Finally, since we assume to have access to labeled datasets $\widehat{\mathcal{S}}_1, \widehat{\mathcal{S}}_2, \ldots, \widehat{\mathcal{S}}_k$ from each source domain and to a unlabeled dataset $\widehat{\mathcal{T}}$ from the target, we may compute empirical estimates of losses (3.28) and (3.29):

$$L_{\text{class}}(\boldsymbol{\beta},\boldsymbol{\theta}_g,\boldsymbol{\theta}_h) \approx -\frac{1}{m} \sum_{j=1}^{k} \frac{\exp(\beta_j)}{\sum_{j'} \exp(\beta_{j'})} \sum_{(\boldsymbol{x},y)\in\widehat{\mathcal{S}}_j^{(m)}} \log p(y \mid \boldsymbol{x};\boldsymbol{\theta}_g,\boldsymbol{\theta}_h), \quad (3.30)$$

*Theoretically, some care should be taken while designing the architecture of the discriminator to make sure that it parametrizes hypotheses in $\mathcal{H}\Delta\mathcal{H}$, but in practice this constraint is dropped. There are similar bounds using the $\mathcal{H}$-divergence instead [91].

$$L_{\text{disc}}(\boldsymbol{\beta}, \boldsymbol{\theta}_g, \boldsymbol{\theta}_d) \approx -\frac{1}{m}\sum_{j=1}^{k}\frac{\exp(\beta_j)}{\sum_{j'}\exp(\beta_{j'})}\sum_{(\boldsymbol{x},y)\in\widehat{\mathcal{S}}_j^{(m)}}\log p(\mathrm{d}=0 \mid \boldsymbol{x}; \boldsymbol{\theta}_g, \boldsymbol{\theta}_d) \tag{3.31}$$
$$-\frac{1}{m}\sum_{\boldsymbol{x}\in\widehat{\mathcal{T}}^{(m)}}\log p(\mathrm{d}=1 \mid \boldsymbol{x}; \boldsymbol{\theta}_g, \boldsymbol{\theta}_d),$$

where $\widehat{\mathcal{S}}_j^{(m)}$ and $\widehat{\mathcal{T}}^{(m)}$ are mini-batches of m examples each from $\widehat{\mathcal{S}}_j$ and $\widehat{\mathcal{T}}$, respectively.

It is instructive to compare our approach with MDAN (Zhao et al. [72]), as that model is the most similar to ours. The bound in equation (3.20) uses one single $\mathcal{H}\Delta\mathcal{H}$-divergence and, therefore, our model comprises one single domain discriminator, which aims to distinguish between the target and the $\boldsymbol{\alpha}$-weighted mixture of source domains. Zhao et al. [72] use k discriminator networks, i.e. one per source domain. More importantly, regarding the choice of $\boldsymbol{\alpha}$, we treat it as one further parameter that can be optimized to minimize the loss. To avoid the objective to collapse into the easiest source domain, as explained in Section 3.4.2.3, we include an extra term that penalizes a sparse $\boldsymbol{\alpha}$. Unlike us, they do not include the sparsity penalization term and choose $\boldsymbol{\alpha}$ to minimize the worst case scenario, by assigning a larger weight to the maximum loss among the k source domains on each training iteration.

3.4.3.2 Consistency regularization on the target domain

Consistency regularization is a key component of many state of the art algorithms in semi-supervised learning. The basic idea is simple: an unlabeled sample and a slightly perturbed version of it should share the same label. Here, this approach is exploited as a sensible heuristic to avoid that domain-invariant features end up hurting the generalization performance of the model on the target domain, as discussed in Section 3.4.2.2.

Specifically, consider a target sample $\boldsymbol{x} \in \widehat{\mathcal{T}}$ and a parametric transformation $\omega : \mathcal{X} \times \Xi \mapsto \mathcal{X}$, where Ξ is the space of parameters for the transformation. Further assume that ω is label-preserving, i.e. $f_{\mathcal{T}}(\omega(\boldsymbol{x}, \xi)) = f_{\mathcal{T}}(\boldsymbol{x})$, $\forall\, \boldsymbol{x} \in \mathcal{X}$, $\xi \in \Xi$. If ω is rich and strong enough, it should spread the transformed target samples over multiple regions of low density under the target distribution, i.e. it will produce new domains. Then, by enforcing agreement on the predictions of original and transformed target samples, we are encouraging the model to learn to extract features that generalize well across domains. Moreover, some of the augmented samples may fall within regions of higher density under the distribution of the source domains. If this hypothesis holds, by enforcing agreement on the predictions of original and transformed target samples and low classification

error on the source domains, we are indirectly promoting low error on the target domain too. Figure 3.19 illustrates this idea.

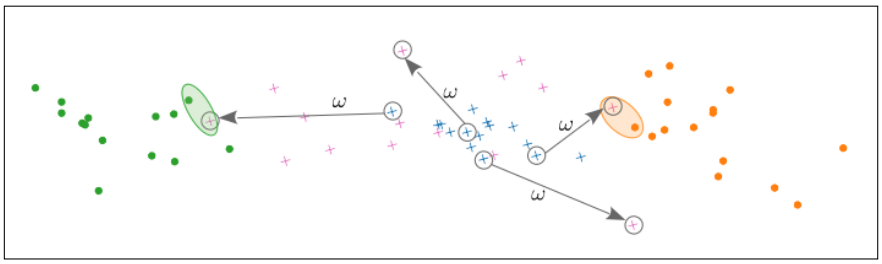


FIGURE 3.19: Toy illustration of the desired effect of the consistency regularization, where images are represented as lying on a 2-D space. Green and orange circles represent (labeled) samples from two distinct source domains; blue and purple x-markers represent original and augmented (unlabeled) target samples, respectively. Colored ellipses enclose pairs of augmented and source samples that are close to each other and therefore are likely to share the same label.

The particular type of consistency regularization we adopt here is FixMatch (Sohn et al. [119]), due to its simplicity and good performance. This approach involves using the predicted class of original (or weakly-transformed) samples as pseudo-labels for the (strongly-)transformed samples and, in our setting, is translated into the following loss function:

$$L_{\text{cons}}(\boldsymbol{\theta}_g, \boldsymbol{\theta}_h) \triangleq -\frac{1}{m} \sum_{\boldsymbol{x} \in \widehat{\mathcal{T}}^{(m)}} \mathbf{1}_{\left(\max_{\mathrm{y} \in \mathcal{Y}}\ p(\mathrm{y}|\boldsymbol{x};\boldsymbol{\theta}_g,\boldsymbol{\theta}_h) > \tau\right)} \log p(\tilde{y} \mid \omega(\boldsymbol{x}, \xi); \boldsymbol{\theta}_g, \boldsymbol{\theta}_h), \tag{3.32}$$

where $\tilde{y} \triangleq \arg\max_{\mathrm{y} \in \mathcal{Y}} p(\mathrm{y} \mid \boldsymbol{x}; \boldsymbol{\theta}_g, \boldsymbol{\theta}_h)$ is the pseudo-label, ξ is chosen randomly for each $\boldsymbol{x}$, and $\tau \geq 0$ is a hyperparameter defining the minimum confidence threshold for the loss to be active. This threshold prevents the loss to be applied too early in the training process and, in our setting, may discard augmented samples that fall too far from the source distributions. The overall objective is then written as follows:

$$\min_{\boldsymbol{\alpha} \in \Delta, \boldsymbol{\theta}_g, \boldsymbol{\theta}_h} \max_{\boldsymbol{\theta}_d} \left\{ L \triangleq L_{\text{class}}(\boldsymbol{\alpha}, \boldsymbol{\theta}_g, \boldsymbol{\theta}_h) + \mu_c \mathcal{L}_{\text{cons}}(\boldsymbol{\theta}_g, \boldsymbol{\theta}_h) - \mu_d \mathcal{L}_{\text{disc}}(\boldsymbol{\alpha}, \boldsymbol{\theta}_g, \boldsymbol{\theta}_d) + \mu_s ||\boldsymbol{\alpha}||_2^2 \right\}, \tag{3.33}$$

where $\mu_c > 0$ controls the relative weight of the consistency loss. A full schematic of our model is provided in Appendix B.2.

Maybe add something about the causal interpretation of this heuristic

It remains to discuss how to build the transformation ω in such a way that it is strong and diverse enough while satisfying the constraint of being label-preserving. This is essentially an application-dependent problem, though. For vision applications, there are multiple simple label-preserving transformations (e.g. translation, rotation, sharpness enhancement, etc.) that can be applied in a pipeline and, as consequence, the transformed

image ends up being a strongly distorted version of the original one. This is the idea followed, for instance, in RandAugment (Cubuk et al. [120]), which we use here. RandAugment receives as input parameters the magnitude and the number of transformations to be applied and randomly chooses the transformations to apply to each sample in the mini-batch. Here, as Sohn et al. [119] did, we choose the number and magnitude of the transformations uniformly at random for each mini-batch.

For generic, non-vision problems, adding random noise sampled from some known distribution (e.g. Gaussian) is a trivial realization of ω. Another possibility is to apply a strong dropout (Srivastava et al. [121]) transformation at the input and inner layers of the neural network. Although such transformation is not necessarily label-preserving, dropout is known to be a successful regularization technique when applied at fully connected layers. We employ this idea in one of the experiments conducted here.

3.4.4 Experiments

3.4.4.1 Experimental protocol

We now conduct several experiments using standard benchmark datasets for multi-source DA. We follow established evaluation protocols for every dataset and, as far as possible, we use the same network architecture for our model and baselines. Notably, comparing with MDAN (Zhao et al. [72]), our single domain discriminator has the same architecture as each of the k domain discriminators in their model. Consequently, our model has less trainable parameters than theirs. Our parameter $\boldsymbol{\beta}$ is initialized uniformly at random in $[0, 1]^k$, so the resulting $\boldsymbol{\alpha}$ initially weights all source domains roughly equally, but it may become sparse as training evolves. We illustrate this behavior in Appendix B.3.4. Hyperparameter tuning was performed through cross-validation over source domains and a hyperparameter sensitivity analysis is conducted in Appendix B.3.3. Further details about the experiments, including the label distributions on each domain (Appendix B.3.1), network architectures (Appendix B.3.5), search ranges for each hyperparameter (Appendix B.3.6), and image transformations (Appendix B.3.7), are also provided as appendices. The PyTorch-based implementation of our model is publicly available.* To facilitate the presentation, from now on we refer to our model as MODA-FM (Multi-source mildly Optimistic Domain Adaptation with FixMatch regularization).

*https://github.com/dpernes/modafm

Baselines DANN-SS (Ganin and Lempitsky [69]): a single-source DA model, where the best results among all source domains are reported. DANN-MS: the same model as before trained on the combined data from all source domains. MADA (Pei et al. [107])I: a state of the art model for single-source DA which tries to align conditional distributions by using one domain discriminator per class (results extracted from Pei et al. [107], we report the best accuracy among all source domains). MDAN (Zhao et al. [72]): a model for multi-source DA, widely described before. MoE (Guo et al. [89]): a state of the art model for multi-source DA that uses one classifier per source domain whose predictions are weighted in an example-dependent way. M^{3}SDA-β (Peng et al. [122]): a moment-matching model for multi-source DA in which two classifiers per source domain are trained to have maximum label discrepancy on the target domain (results extracted from Peng et al. [122]). Fully supervised: fully supervised model trained on the target data, to provide an empirical upper bound on the performance of the DA task. MODA: our model without consistency regularization (i.e. $\mu_c = 0$). FM: our model without domain discriminator, trained on the naively combined data from all source domains and using FixMatch consistency regularization as described in Section 3.4.3.2.

Digits classification In this experiment, the task is digit classification using 4 datasets: MNIST (LeCun et al. [123]), MNIST-M (Ganin and Lempitsky [69]), SVHN (Netzer et al. [124]), and SynthDigits (Ganin and Lempitsky [69]). We take each of the first three datasets as the target in turn, and use the remaining as source domains. The number of training images chosen randomly from each domain, including the target, is 20k. The evaluation is performed in the non-transductive setting, i.e. no target data used during training are used for evaluation. The results are in Table 3.6.

	Digits				Office-31			
	MNIST	MNIST-M	SVHN	Avg.	Amazon	DSLR	Webcam	Avg.
DANN-SS [69]	97.9 ±0.4	73.2±1.6	72.8 ±3.3	81.3	60.5 ±1.4	**100.0** ±0.0	98.0 ±0.3	86.2
DANN-MS [69]	97.9 ±1.6	67.5 ±1.4	71.5 ±1.6	79.0	61.2 ±1.3	99.9 ±0.2	98.8 ±0.3	86.6
MADA [107]	–	–	–	–	70.3 ±0.3	99.6 ±0.1	97.4 ±0.1	89.1
MDAN [72]	98.3 ±0.2	69.1 ±1.2	69.5 ±2.8	79.0	65.2 ±0.4	99.3 ±0.2	97.8 ±0.5	87.4
MoE [89]	98.6 ±0.2	69.9 ±1.1	81.8 ±0.8	83.4	–	–	–	–
MODA	98.4 ±0.2	77.4 ±1.6	71.7 ±1.5	82.5	65.5 ±0.5	99.9 ±0.2	99.0 ±0.3	88.1
FM	**99.2** ±0.1	91.1 ±0.4	**90.0** ±0.8	93.4	70.3 ±0.6	99.7 ±0.2	**99.2** ±0.4	89.7
MODA-FM	98.8 ±0.1	**95.4** ±0.4	89.4 ±1.4	**94.5**	**70.7** ±0.9	**100.0** ±0.0	99.1 ±0.1	**89.9**
Fully supervised	98.9 ±0.1	96.2 ±0.1	90.3 ±0.5	95.1	88.1 ±1.6	99.3 ±1.1	99.5 ±0.7	95.6

TABLE 3.6: Average accuracy ± standard deviation (%) over 5 independent runs on digits and objects classification (Office-31). The domain on each column corresponds to the target.

Object classification on Office-31 dataset Office-31 (Saenko et al. [125]) is a standard benchmark dataset for domain adaptation. It comprises 31 object categories extracted from 3 domains: Amazon, which contains 2817 images downloaded from amazon.com, DSLR and Webcam, which contain 498 and 795 images captured with DSLR cameras and webcams, respectively, under different environments. In this experiment, we adopt the fully-transductive setting, following Pei et al. [107], where all unlabeled data from the target domain are used for training (except for the fully supervised model, where we use 80% of the data for training and the remaining for testing). All models are implemented using a pre-trained (on ImageNet) ResNet-50 (He et al. [126]) as the base architecture. We take each domain as the target in turn and all remaining are used as sources. The results are presented in Table 3.6.

Sentiment analysis on Amazon Reviews dataset The Amazon Reviews dataset (Blitzer et al. [127]) is another multi-domain dataset widely used as a benchmark for domain adaptation. It contains binary (i.e. positive and negative) reviews on four types of products: books, DVDs, electronics, and kitchen appliances. Here, we follow the experimental setting from Chen et al. [128], where samples were pre-processed to 5k-dimensional TF-IDF feature vectors, thus word order information was not preserved. We choose 2k training samples from each domain randomly and the remaining target samples are used for testing, following the non-transductive setting. Since the data are not images, we use dropout as our (pseudo-)label-preserving transformation. Specifically, on each training iteration, we randomly choose a dropout rate (in a pre-specified range) to be applied at the input and hidden layers of the multi-layer perceptron (MLP) and the corresponding output prediction is used as the pseudo-label $\tilde{y}$. No dropout is applied for source and non-augmented target samples. All domains are taken as the target in turn and all remaining are used as sources. The results are in Table 3.7.

Large scale object classification on DomainNet dataset To the best of our knowledge, the DomainNet dataset (Peng et al. [122]) is the largest domain adaptation image dataset to date. It consists of 6 domains, each containing 345 categories of common objects. The domains are: clipart images (*clp*), infographic images (*inf*), artist paintings (*pnt*), drawings made by worldwide players of the game "Quick, Draw!" (*qdr*), real world images (*rel*), and sketches of objects (*skt*). The dataset consists of around 423.5k images split in prespecified train/test partitions. For training the models, we use all the data from the

	Books	DVD	Electronics	Kitchen	Avg.
DANN-SS [69]	77.5 ±1.0	78.9 ±0.9	84.2 ±0.2	85.8 ±0.4	81.6
DANN-MS [69]	78.9 ±0.2	80.8 ±1.0	84.7 ±0.6	87.0 ±0.4	82.9
MDAN [72]	79.1 ±0.3	81.3 ±0.8	84.6 ±0.3	85.6 ±1.6	82.7
MoE [89]	80.3 ±0.3	81.9 ±0.6	85.2 ±0.6	87.4 ±0.5	83.7
MODA	79.0 ±0.2	80.7 ±1.2	84.7 ±0.3	86.8 ±0.6	82.8
FM	78.8 ±2.0	81.0 ±1.5	**85.6** ±0.9	87.6 ±0.7	83.3
MODA-FM	**80.9** ±1.1	**82.0** ±1.0	85.3 ±0.4	**88.4** ±0.3	**84.2**
Fully supervised	83.6 ±0.4	83.6 ±0.6	85.4 ±0.4	87.8 ±0.2	85.1

TABLE 3.7: Average accuracy ± standard deviation (%) over 5 independent runs on sentiment analysis (Amazon Reviews). The domain on each column corresponds to the target.

training partition, excluding the labels for the target domain. For testing, we use the target domain data from the test partition. All models were implemented on top of a pre-trained (on ImageNet) ResNet-152. All domains are taken as the target in turn and all remaining are used as sources. The results are in Table 3.8.

	clp	*inf*	*pnt*	*qdr*	*rel*	*skt*	Avg.
DANN-MS [69]	**62.6**	21.7	50.3	14.3	**63.5**	48.3	43.5
MDAN [72]	52.2	19.1	46.0	12.6	48.2	40.4	36.4
M^{3}SDA-β [122]	58.6	**26.0**	**52.3**	6.3	62.7	49.5	42.6
MODA-FM	61.2	24.5	51.2	**17.2**	62.3	**52.4**	**44.8**
Fully supervised	68.5	28.7	61.5	64.5	78.5	59.3	60.2

TABLE 3.8: Accuracy on object classification (DomainNet). The domain on each column corresponds to the target.

3.4.4.2 Discussion

Tables 3.6, 3.7, and 3.8 show that our model outperforms the baselines in most settings and performs comparably to the fully-supervised model in many. When no consistency regularization is used (MODA), our approach exhibits a higher accuracy than DANN-SS, DANN-MS and MDAN in most cases. This observation shows that our mildly optimistic combination of source domains works better than using only the data from the best source domain (DANN-SS) or naively combining the data from all source domains (DANN-MS). It also suggests that MDAN wastes too much computational effort on optimizing itself for the hardest source domain. MADA and MoE perform better than our non-regularized model in some cases, which is not surprising since both methods try to mitigate somehow the curse of domain-invariant representations. Their advantage is, therefore, mostly noticeable when the target shift is large (e.g. SVHN and all domains in Office-31 – see Appendix B.3.1), but the performance is still below MODA-FM.

Very significant performance gains are observed when we apply the consistency regularization, particularly in the visual datasets (digits and Office-31). These gains are more expressive in the most challenging settings, i.e. when the target data are perceptually very different from the source data (e.g. MNIST-M – see Appendix B.3.8) or when the target shift is large. The latter observation suggests that this regularization succeeds on mitigating the curse of domain-invariant representations, as hypothesized before. This is strongly corroborated by an experiment we present in Appendix B.3.2, where we show that the model accuracy on the target domain keeps stably high when the model is trained for a large number of epochs. Interestingly, though, we observe that, in some settings, FM outperforms MODA-FM, although the differences are small. In these cases domain-invariant representations are slightly hurting the performance and/or the source weights α are sub-optimal. Besides being the largest dataset, DomainNet is also the most challenging for the DA task. For almost all domains, there is a large gap between the performance of all DA models (including ours) and the fully supervised. Interestingly, none of the multi-source DA algorithms does much better than the naive DANN-MS for this particular dataset. This might be explained by the large dissimilarity across domains in DomainNet: since no source domain is particularly close to the target domain, weighting all source domains equally works ends up working fairly well. Nonetheless, our model achieves the best average performance across all source domains. Finally, it is worth highlighting the positive effect provided by using dropout as the label-preserving transformation ω in the experiment with Amazon Reviews. This observation suggests that this methodology can be applied successfully to non-visual data too.

3.4.5 Conclusion

We have presented a novel algorithm that achieves state of the art results in unsupervised multi-source domain adaptation. In our approach, the problem is formulated as DA from a single source domain whose distribution corresponds to a mixture of the original source domains. The mixture weights are adjusted dynamically throughout the training process, according to a mildly optimistic objective. Additionally, we employ FixMatch on the target samples, a form of consistency regularization that proves to have a strong impact on the model performance and to be capable of mitigating the curse of domain-invariant representations. This regularization relies on a label-preserving transformation, which is hard to construct for non-visual data. Moreover, better results could be achieved if both

the label-preserving transformation and the source weights were learned to approximate the augmented target samples to the source samples. Both problems are interesting lines for future research.

3.5 Summary

In this chapter, we addressed the problem of domain adaptation, a particular type of transfer learning where the label space $\mathcal{Y}$ is constant across domains (i.e. the learning task is the same on source and target domains). Contrary to Chapter 2, where the goal was to exploit correlations in data from multiple entities to build a model with good performance across all training domains, in DA the focus is on a specific entity/domain for which no labeled data is available.

In Section 3.3, we investigated several strategies to combine a multi-source DA algorithm with a CNN for object counting in videos. The purpose was to build a robust model capable of generalizing well to new cameras with different points of view, without the need to collect further annotated data. Moreover, this was one of the first few attempts to extend the success of adversarial DA techniques to temporal data and, in particular, to video sequences, a topic that has been seldom explored in the literature.

In Section 3.4, we presented our novel algorithm for unsupervised multi-source DA, the main contribution of this chapter. This algorithm builds upon existing methods by i) learning to weight source domains jointly with the minimization of source error and alignment of marginal distributions and ii) employing a regularization technique that proved to successfully mitigate the curse of domain-invariant representations, particularly for image data. The latter had a highly beneficial effect on the performance and was therefore crucial for the state of the art results achieved by this model.

Chapter 4

Domain generalization

Some parts of this chapter were originally published in or adapted from:

> [118] P. M. Ferreira, D. Pernes, A. Rebelo, and J. S. Cardoso, "DeSIRe: Deep signer-invariant representations for sign language recognition," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019 (presented in Section 4.5)
>
> [129] P. M. Ferreira, D. Pernes, A. Rebelo, and J. S. Cardoso, "Learning signer-invariant representations with adversarial training," in *Twelfth International Conference on Machine Vision (ICMV 2019)*, vol. 11433. International Society for Optics and Photonics, 2020, p. 114333D (presented in Section 4.3)
>
> [130] ——, "Signer-independent sign language recognition with adversarial neural networks," *International Journal of Machine Learning and Computing*, vol. 11, no. 2, 2021 (idem)
>
> [131] P. M. Ferreira, A. F. Sequeira, D. Pernes, A. Rebelo, and J. S. Cardoso, "Adversarial learning for a robust iris presentation attack detection method against unseen attack presentations," in *2019 International Conference of the Biometrics Special Interest Group (BIOSIG)*, 2019, pp. 1–7 (Section 4.4)

The first two authors contributed equally in [118] and [129]. Both conceived the models and designed and conducted the experiments, with the supervision of Rebelo and Cardoso. Diogo Pernes was slightly more focused on the problem and model formalization, while Ferreira was more involved on dataset selection and preprocessing. The work in [130] extends [129] by including a more exhaustive experimental evaluation. In [131], Diogo Pernes contributed on the development of the proposed methodology, together with the first two authors, who motivated the application, formalized the problem and conducted all experiments. Rebelo and Cardoso supervised the work.

4.1 Introduction

In Chapters 2 and 3, the target entities/domains were known at training time. In Chapter 2, we exploited the correlations between different but related entities to augment the amount of data available for each of those and hence improve the in-distribution generalization. Chapter 3 was dedicated to the problem of domain adaptation, whose purpose is to improve the out-of-distribution (OOD) generalization in a specific target domain for which no labeled data is available.

In this chapter, we shall continue focusing on OOD generalization. However, now, the target domain is unknown and, therefore, no data from this domain is available at training time, neither labeled nor unlabeled. The purpose, then, is to use labeled data from multiple source domains to build a discriminative model that generalizes well to unknown OOD target domains – a problem known as *domain generalization* (Blanchard et al. [132], Muandet et al. [133]). Our main assumption to accomplish this goal is that the set of features that are relevant for the learning task are domain-invariant. Formally, we assume that, for each domain $\mathcal{D}$, there exists a bijection $b_{\mathcal{D}} : \mathcal{X} \mapsto \mathcal{Z} \times \mathcal{W}$, where $\mathcal{Z}$ is the domain-invariant space of features used for classification and $\mathcal{W}$ are domain-specific auxiliary features carrying no relevant signal for the desired learning task. Thus, for $(\mathrm{z}, \mathrm{w}) \triangleq b_{\mathcal{D}}(\mathbf{x})$, we assume that $p_{\mathcal{D}}(\mathrm{y} \mid \mathrm{x}) = p(\mathrm{y} \mid \mathrm{z})$, i.e. the optimal classifier for any domain $\mathcal{D}$ can be reconstructed from features in $\mathcal{Z}$ and a domain-invariant classifier $p(\mathrm{y} \mid \mathrm{z})$. This formulation is closely related to the covariate shift assumption for domain adaptation, described in Section 3.2.2.4.

A computer vision application where this problem is particularly relevant is sign language recognition (SLR). Large inter-signer variability in the manual signing process of sign languages is one of the challenges associated with this task. Due to this issue, models trained on data from a given set of signers often fail to generalize well when tested on previously unseen signers. Since, ideally, an SLR system should be able to recognize the gestures of any signer, this problem should be tackled with domain generalization (DG) techniques. For this reason, SLR will be the main application considered in this chapter. Nonetheless, we will also show that the same principles can be applied successfully to develop a fingerprint presentation attack detection method that exhibits robust performance on detecting unseen attacks.

The remainder of this chapter is organized as follows: i) we start by reviewing the state of the art for DG (Section 4.2); ii) we present a novel adversarial-based approach for DG

in the context of SLR (Section 4.3); iii) we show how this methodology can be successfully adapted to address the problem of iris presentation attack detection (Section 4.4); iv) we present a novel reconstruction-based algorithm for DG (Section 4.5).

4.2 State of the art

Zhou et al. [134] divide the algorithms for domain generalization as heterogeneous and homogeneous, depending on whether the label space varies (heterogeneous DG) or not (homogeneous DG). The former case is also known as *zero-shot* domain generalization and its goal is, in general, to learn a feature representation that can be used in the target domain to recognize new classes. The latter, which will be the focus of this chapter, is closely related to domain adaptation, so there is a significant intersection between the two. Albuquerque et al. [135] presented an upper bound for the generalization error that is essentially an upper bound for multi-source domain adaptation, similar to the bound by Zhao et al. [72] (Theorem 3.3) and to our own (Theorem 3.4).

The theoretical proximity between the two problems motivates the existence of similar algorithms to tackle them. Many algorithms for DG actually follow the paradigm of domain alignment, which we have discussed extensively in the context of DA. Li et al. [136] use an adversarial autoencoder and maximum mean discrepancy over its latent space to learn domain-invariant features. Ghifary et al. [137] address the same problem through a multi-output autoencoder, which is trained to transform samples from one domain into samples from the remaining domains with the same label. Motiian et al. [138] proposed a unified framework to address the problems of domain adaptation and generalization. They use a contrastive L^2-loss in the latent space that pushes together samples from different domains and the same class while pulling apart samples from different classes. Several other approaches extend the idea of domain adversarial networks (Ganin and Lempitsky [69]) to the problem of domain generalization, by using domain classifiers and minimax training to learn domain-invariant features. Some of those use a single multi-class classifier to classify samples into one of k source domains (e.g. Aslani et al. [139], Matsuura and Harada [140]) and others employ k binary domain discriminators trained in a one-vs-all manner (e.g. Shao et al. [141], Li et al. [142]).

Ensemble learning has also been widely applied to the problem of domain generalization. Xu et al. [143] train support vector machines (SVMs) with a single positive example

and a few negative examples (known as *exemplar-SVMs*) and use the most confident classifiers in an ensemble to make the final prediction. More recent approaches replace the SVM with deep neural networks and build ensembles of domain-specific networks, either by weighting all the predictions equally (e.g. D'Innocente and Caputo [144], Zhou et al. [145]) or by using the output of a domain classifier as sample-dependent ensemble weights (Wang et al. [146]).

Self-supervised learning (SSL) techniques are becoming increasingly popular in machine learning and have also been applied to the problem of DG. SSL refers to the task of learning from free labels, i.e. it consists of standard supervised pretraining on tasks where the labels can be extracted automatically from the data, without the need of manual annotation. Examples of SSL tasks are predicting the next word in a sentence, image colorization (Zhang et al. [147]), predicting the relative position of image patches (Doersch et al. [148]), predicting if a video is being played forward or backward (Wei et al. [149]), etc. The idea motivating SSL is that the features learned by pretraining the model on self-supervised tasks provide good initializations for the model, which can then be finetuned for the desired task using a smaller amount of annotated data. In the scope of DG, SSL provides useful features regardless of the target task, reducing the overfitting to domain-specific biases (Zhou et al. [134]). This idea was followed by Carlucci et al. [150] and Wang et al. [151], who trained a network to solve the Jigsaw puzzle (i.e. to place nine shuffled image patches back into their correct positions) as an auxiliary task to enhance domain generalization.

For a more complete review of DG theory and algorithms, please see Wang et al. [152] and Zhou et al. [134].

4.3 Adversarial domain generalization for signer-independent sign language recognition

4.3.1 Introduction

Sign language is an integral form of communication and, currently, considered the standard education method of deaf people worldwide. It is a visual means of communication, with its own lexicon and grammar, that combines articulated hand gestures along with facial expressions to convey meaning. Deaf people have difficulty in speaking and learning spoken languages like hearing people. However, with sign language, they are able to

communicate as efficiently and seamlessly. The population of sign language speakers is extended to family and friends of the deaf, interpreters, and those who learn the language by their own initiative. As most hearing people are unfamiliar with sign language, deaf people find it difficult to interact with the hearing majority. The result is the isolation of deaf communities from the overall society.

In this regard, automatically analyzing and recognizing sign language has become one of the key problems in the human computer-interaction field. SLR systems are meant to automatically translate signs into the corresponding text or speech. This is important not only to bridge the communication gap between deaf and hearing people but also to increase the amount of content the deaf can access, such as the creation of educational tools or games for deaf people and visual dictionaries of sign language.

The SLR problem has been addressed in the literature by means of wearable devices (e.g. data gloves or similar equipments) or vision-based systems (Ebrahim Al-Ahdal and Nooritawati [153]). Vision-based systems, either those using color or depth information, face the problem of the inherently noisy and ambiguous nature of the input data. Data gloves yield more reliable and descriptive features. Nevertheless, vision-based SLR systems are arguably the most natural choice for real-world applications. Vision-based SLR is less invasive since there is no need to wear cumbersome devices that may affect the natural signing movement.

Several vision-based SLR methodologies have been proposed over the last twenty years, with increasing progress in the recognition performance. An important part of this recent progress was achieved thanks to the emergence of deep learning approaches and more specifically with CNNs (Pigou et al. [154], Koller et al. [155], Wu et al. [156], Neverova et al. [157], Kumar et al. [158]).

A practical SLR system must operate in a signer-independent scenario. That is, the signer of the probe must not be seen during the training process of the models. Although current SLR systems demonstrate excellent performance for signer-dependent settings, their recognition rates typically decrease significantly when the signer is new to the system. This performance drop is the result of the large inter-signer variability in the manual signing process of sign languages.

Although the appearance of manual signs is well-defined in sign language dictionaries, in practice, variations may arise due to regional and social factors, and also from age,

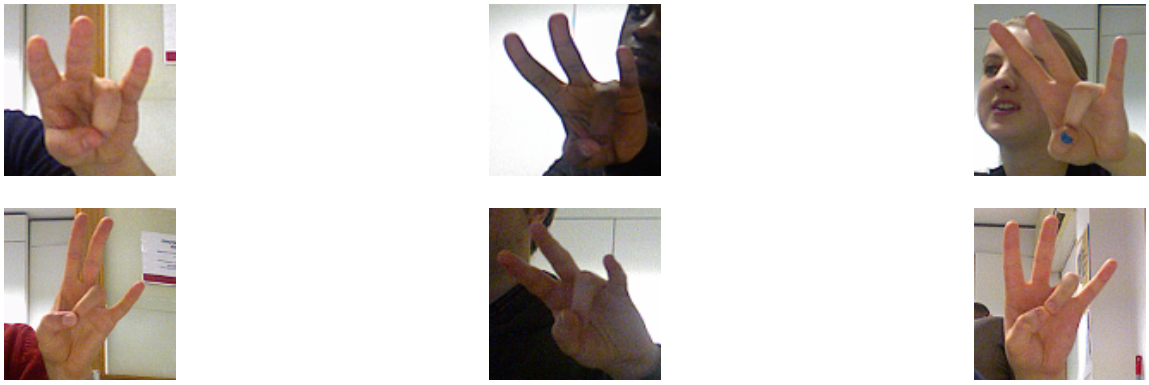


FIGURE 4.1: Inter-signer variability: it is possible to observe not only phonological variations (e.g. different handshapes, palm orientations, and sign locations) but also a large physical variability (e.g. different hand sizes) when six signers are performing the same sign.

gender, education and family background. This can lead to significant variations in manual signs performed by different signers, and pose challenging problems for developing robust signer-independent SLR systems. Figure 4.1 illustrates inter-signer variability by showing six different signers performing the same gestures.

Borrowing from recent works on adversarial neural networks (Goodfellow et al. [159], Feutry et al. [160]) and domain transfer (Ganin and Lempitsky [69]), we introduce a deep neural network along with a novel adversarial training objective to specifically tackle the signer-independent SLR problem. The underlying idea is to preserve as much information as possible about the signs, while discarding the signer-specific information that is implicitly present in the manual signing process. For this purpose, the proposed deep model is composed by an *encoder* network, which maps from the input images to latent representations, as well as two discriminative classifiers operating on top of these underlying representations, namely the *sign-classifier* network and the *signer-classifier* network. While the former is trained to predict the sign labels, the latter is trained to identify the signer. In addition, the parameters of the encoder network are optimized to minimize the loss of the sign-classifier while trying to fool the signer-classifier network. This adversarial and competitive training scheme encourages the learned representations to be signer-invariant and highly discriminative for the sign classification task. To further constrain the latent representations to be signer-invariant, we introduce an additional training objective that operates on the hidden representations of the encoder network in order to enforce the latent distributions of different signers to be as similar as possible.

Although this adversarial training framework is similar to those initially introduced by Ganin and Lempitsky [69], in the context of domain adaptation, and then by Feutry

et al. [160] to learn anonymized representations, our main contributions on top of these works are two-fold: i) the application of the adversarial training concept to the signer-independent SLR problem and ii) a novel adversarial training objective that differs from the ones of Ganin and Lempitsky [69] and Feutry et al. [160] in two ways. First, our training objective is minimum if and only if the adversarial classifier (i.e. the signer-classifier) produces a uniform distribution over the domains (i.e. signer identities). Second, we introduce an additional term to the adversarial training objective that further discourages the learned representations of retaining any signer-specific information, by explicitly imposing similarity in the latent distributions of different signers.

The remainder of this section is organized as follows. Section 4.3.2 presents the related work on SLR. The proposed model along with its adversarial training scheme are fully described in Section 4.3.3. Experimental results and conclusions are reported in Sections 4.3.4 and 4.3.5, respectively.

4.3.2 Related Work

We have discussed some of the most relevant approaches for DG in Section 4.2, so now we shall focus our attention on the specific problem of SLR. This has become an appealing topic in modern society because such systems can ideally be used to reduce the communication barriers that exist between deaf and hearing people. SLR approaches can be broadly divided into: (i) isolated, which address the recognition of single signs either using static images or video (Marin et al. [161, 162]), and (ii) continuous, which correspond to the recognition of sentences represented as a sequence of signs (Guo et al. [163, 164], Wang et al. [165]). Although most recent works focus on the continuous SLR and its associated problems (e.g. large vocabulary size), static SLR is still a challenging task, especially under unconstrained scenarios. One of the biggest challenges is related to the large inter-signer variability, which is in fact the focus of this work.

According to the amount of data required from the test signers, previously signer-independent SLR works can be broadly divided into two main groups: (i) signer adaptation approaches, where a previous trained model is adapted to a new signer by using a small amount of signer specific data, and (ii) truly signer independent methodologies, in which a generic model robust for new test signers is built without using data of those test signers.

The former signer adaptation approaches were greatly inspired by speaker adaptation methods from the speech recognition research. von Agris et al. [166] used maximum likelihood linear regression (MLLR) and maximum a posteriori (MAP) estimation for signer adaptation. In a subsequent work (von Agris et al. [167]), they extended their work by combining the eigenvoice approach by Kuhn et al. [168] with MLLR and MAP to adapt trained hidden Markov models (HMMs) to new signers. MLLR and MAP were the basic adaptation strategies, and the eigenvoice approach provided constraints to reduce the number of free parameters to be adapted. More recently, Kim et al. [169] investigated the potential of several signer normalization techniques (e.g. speed normalization) and different deep neural network adaptation strategies for the signer-independence problem. They found that while signer normalization is ineffective, a simple neural network adaptation strategy, such as fine-tuning the signer-specific neural networks on the adaptation data, is very effective.

The aforementioned methods are all supervised adaptation approaches, in the sense that the adaptation data from the new signer must be labeled. However, in practice, collecting labeled data may be a cumbersome and time-consuming task. To overcome this issue, a few works have resorted to unsupervised adaptation strategies. Yin et al. [170] proposed a two-step weakly supervised metric learning framework to perform signer adaptation with some unlabeled sign data of the new signer. In the first step, a generic metric is learnt from the available labeled data of several different signers. In the second step, the generic metric is adapted to the new signer by considering clustering and manifold constraints along with the collected unlabeled data.

Although signer adaptation is a reasonable approach, there is still the need to collect either labeled or unlabeled data to retrain and adapt the model to a new signer. Therefore, a truly signer-independent approach, which does not require any data from the new signers, would be the ideal solution for a practical SLR system. Examples of such works can be found are those by Zieren and Kraiss [171], Shanableh and Assaleh [172], von Agris et al. [173], Kong and Ranganath [174], Kelly et al. [175], Dahmani and Larabi [176], Yin et al. [177]. Most of them involved a huge feature engineering effort in order to build normalized feature descriptors robust to the physical variations of the signers (e.g. height, hand size and length of the arm) and different acquisition conditions (e.g. distance to the camera). Afterwards, most of these works use HMMs or their variants for sign recognition. It is the example of the work proposed by von Agris et al. [173], in which a set of

11 regional features are extracted (e.g. 2-D coordinates, hand blobs area, orientation of the main axis, inertia ratio, eccentricity and compactness) and then normalized according to the head position and shoulders distance of the signer. Kelly et al. [175] introduced a novel signer-independent hand posture feature descriptor, along with an eigenspace size function which represents both qualitative and quantitative properties of a visual shape. Kong and Ranganath [174] gave particular importance to the movement of epenthesis (ME), which appears as the transition movement that connects successive signs. Concretely speaking, they removed the ME by using a segment and merge approach to decrease the inter-signer variations in ME and used a two-layer conditional random field classifier for sign recognition. More recently, Yin et al. [177] proposed an interesting and alternative approach that relies on distance metric learning. In particular, the metric is learnt by constraining the distances between the training samples and generic references of the sign classes. The references are constructed by signer invariant representations of each sign class (i.e. the average of all samples within the specific class). Afterwards, a two-step iterative optimization strategy is employed to obtain more appropriate references and update the corresponding distance metric alternately.

Although the aforementioned methods have promoted a significant evolution in the signer-independent research, there are still many opportunities for improvement. A major weakness across all the methods is related to the fact that representation and metric learning are not performed jointly. It is well known that the recent success of deep learning approaches, particularly those using CNNs, in tasks like object detection and recognition, has been extended to the SLR problem. The underlying motivation is to automatically learn multiple levels of representations directly from the data (Pigou et al. [154], Koller et al. [155], Wu et al. [156], Neverova et al. [157], Kumar et al. [158]). However, none of these explicitly constrains the learned representations to be signer invariant.

4.3.3 Methodology

The ultimate goal of our model is to learn signer-invariant latent representations that preserve the relevant part of the information about the signs while discarding the signer-specific traits that may hamper the sign classification task. To accomplish this purpose, we introduce a deep neural network along with an adversarial training scheme that is able to learn feature representations that combine both sign discriminativeness and signer-invariance.

More specifically, let $\{(\boldsymbol{X}_i, y_i, s_i)\}_{i=1}^{n}$ be a labeled dataset of n samples, where $\boldsymbol{X}_i$ represents the i-th colour image, and $y_i \in \mathcal{Y}$ and $s_i \in \{1, 2, \ldots, k\}$ denote the corresponding class (sign) label and signer identity, respectively. To induce the model to learn signer-invariant representations, the proposed model comprises three distinct sub-networks:

- an encoder network, which aims at learning an encoding function $g(\cdot; \boldsymbol{\theta}_g) : \mathcal{X} \mapsto \mathcal{Z}$, parameterized by $\boldsymbol{\theta}_g$, that maps from an input image $\boldsymbol{X} \in \mathcal{X}$ to a latent representation $\boldsymbol{z} \in \mathcal{Z}$;

- a sign-classifier network, which operates on top of this underlying latent representation $\boldsymbol{z}$ to learn our task-specific function $h(\cdot; \boldsymbol{\theta}_h) : \mathcal{Z} \mapsto \mathcal{Y}$, parameterized by $\boldsymbol{\theta}_h$, that maps latent vectors into the predicted probabilities of each sign class;

- a signer-classifier network, with the purpose of learning a signer-specific function $d(\cdot; \boldsymbol{\theta}_d) : \mathcal{Z} \mapsto \{1, 2, \ldots, k\}$, parameterized by $\boldsymbol{\theta}_d$, that maps the same hidden representation $\boldsymbol{z}$ into the corresponding signer identity.

During the learning stage, the parameters of both classifiers are optimized in order to minimize their errors on their specific tasks on the training set. In addition, the parameters of the encoder network are optimized in order to minimize the loss of the sign-classifier network while forcing the signer-classifier to be a random guessing predictor. In the course of this adversarial training procedure, the learned latent representations $\boldsymbol{z}$ are encouraged to be signer-invariant and highly discriminative for sign classification. To further discourage the latent representations of retaining any signer-specific traits, we introduce an additional training objective that enforces the latent distributions of different signers to be as similar as possible.

4.3.3.1 Architecture

As illustrated in Figure 4.2, the architecture of the proposed model is composed by three main sub-networks or blocks, i.e. an encoder, a sign-classifier, and a signer-classifier.

The encoder network attempts to learn a mapping from an input image $\boldsymbol{X}$ to a latent representation $\boldsymbol{z}$. It consists of a sequence of three pairs of consecutive 3×3 convolutional layers with Rectified Linear Units (ReLUs) as non-linearities. For downsampling, the last convolutional layer of each pair has a stride of 2. The number of filters starts as 32 and is doubled after each convolutional pair. The dense layer on top of the encoder network has

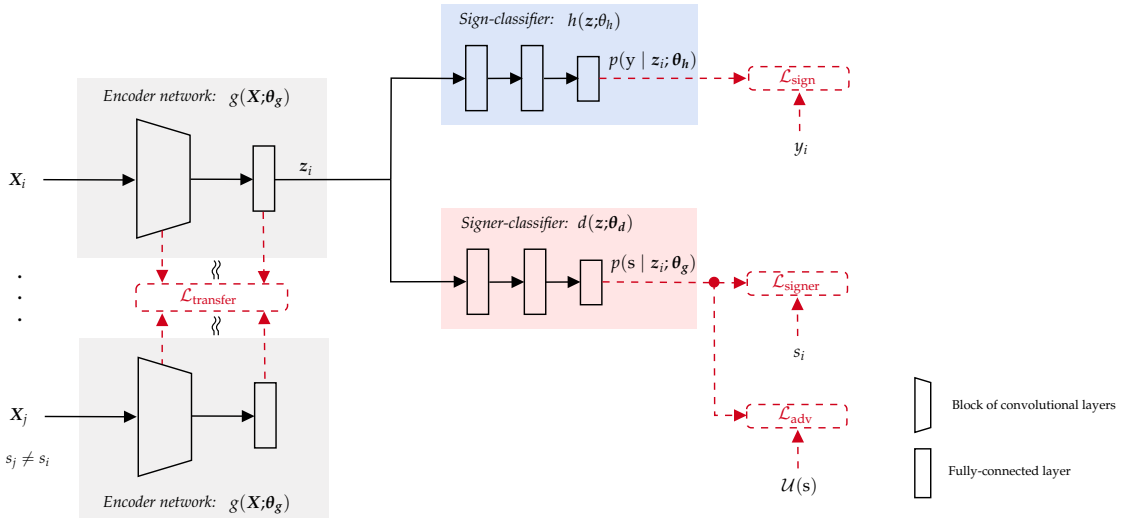


FIGURE 4.2: The architecture of the proposed signer-invariant neural network. It comprises three main sub-networks or blocks, i.e. an encoder, a sign-classifier and a signer-classifier.

128 neurons. On top of that, there is a fully-connected layer, also with a ReLU, outputting the desired signer-invariant latent representations z.

Taking the latent representations z as input, the sign-classifier block is composed by a sequence of three fully-connected layers, with ReLUs as the non-linear functions, for predicting the sign class $\hat{y} \triangleq h(z;\theta_h)$. The number of nodes of each hidden layer was set to 128. The last fully-connected layer has a softmax activation function which outputs the probabilities for each sign class.

The signer-classifier network has exactly the same topology as the sign-classifier. However, it maps the latent representations z to the predicted signer identity $\hat{s} \triangleq d(z;\theta_d)$. Therefore, the number of nodes of the output layer equals the number of signers in the training set.

4.3.3.2 Adversarial training

By definition, signer-invariant representations discard all signer-specific information and, as such, no function (i.e. classifier) exists that maps such representations into the correct signer identity. This naturally leads to an adversarial problem, in which: i) a signer-classifier network $d(\cdot;\theta_d)$ receives latent representations $z \triangleq g(X;\theta_g)$ from an encoder network $g(\cdot;\theta_g)$ and tries to predict the signer identity s corresponding to image X and ii) the encoder network tries to fool the signer-classifier network while still providing good

representations for the sign-classifier network $h(\cdot;\boldsymbol{\theta}_h)$, which in turn receives the same representations $\boldsymbol{z}$ and aims to predict the sign label y corresponding to image $\boldsymbol{X}$.

Therefore, the signer-classifier network shall be trained to minimize the negative log-likelihood of correct signer predictions:

$$\min_{\boldsymbol{\theta}_d} \left\{ L_{\text{signer}}(\boldsymbol{\theta}_g, \boldsymbol{\theta}_d) \triangleq -\frac{1}{n} \sum_{i=1}^{n} \log p(s_i \mid g(\boldsymbol{X}_i; \boldsymbol{\theta}_g); \boldsymbol{\theta}_d) \right\} \tag{4.1}$$

In the perspective of the encoder, the predictions of the sign-classifier should be as accurate as possible and the predictions of the signer-classifier should be kept close to uniform, meaning that this latter model is not capable of doing better than random guessing the signer identity. Formally, this may be translated into the following constrained objective:

$$\min_{\boldsymbol{\theta}_g, \boldsymbol{\theta}_h} \left\{ L_{\text{sign}}(\boldsymbol{\theta}_g, \boldsymbol{\theta}_h) \triangleq -\frac{1}{n} \sum_{i=1}^{n} \log p(y_i \mid g(\boldsymbol{X}_i; \boldsymbol{\theta}_g); \boldsymbol{\theta}_h) \right\}, \tag{4.2}$$

$$\text{subject to } \frac{1}{n} \sum_{i=1}^{n} D_{\text{KL}}(\mathcal{U}(\text{s}) || p(\text{s} \mid g(\boldsymbol{X}_i; \boldsymbol{\theta}_h); \boldsymbol{\theta}_g) \leq \epsilon, \tag{4.3}$$

where D_{KL} is the Kullback-Leibler (KL) divergence and $\mathcal{U}(\text{s})$ denotes the discrete uniform distribution on the random variable s, defined over the set of signer identities $\{1, 2, \ldots, k\}$ in the training set. Here, $\epsilon \geq 0$ determines how far from uniform the signer-classifier predictions are allowed to be (as measured by the KL divergence). The choice of the uniform distribution implies the underlying assumption that the training set is balanced relatively to the number of examples per signer (which should be true for most practical datasets). When this is not the case, the empirical distribution of signer identities in the training set may be used instead.

The inequality constraint (4.3) may be rewritten as:

$$L_{\text{adv}}(\boldsymbol{\theta}_g, \boldsymbol{\theta}_d) \triangleq \frac{1}{nk} \sum_{i=1}^{n} \sum_{\text{s}} \log p(\text{s} \mid g(\boldsymbol{X}_i; \boldsymbol{\theta}_g); \boldsymbol{\theta}_d) \leq \epsilon + \log k, \tag{4.4}$$

and the constrained optimization problem may be equivalently formulated as:

$$\min_{\boldsymbol{\theta}_g, \boldsymbol{\theta}_h} \left\{ L(\boldsymbol{\theta}_g, \boldsymbol{\theta}_h, \boldsymbol{\theta}_d) \triangleq L_{\text{sign}}(\boldsymbol{\theta}_g, \boldsymbol{\theta}_h) + \mu_d L_{\text{adv}}(\boldsymbol{\theta}_g, \boldsymbol{\theta}_d) \right\}, \tag{4.5}$$

where $\mu_d \geq 0$ depends on ϵ and L_{adv} plays the role of an adversarial loss with respect to the signer classification loss L_{signer}.

This objective and the structure of our model are similar to those used by Ganin and Lempitsky [69], in the context of domain adaptation, and by Feutry et al. [160], to learn

anonymized representations for privacy purposes. However, the former uses the negative signer classification loss as the adversarial term (i.e. $L_{\text{adv}} \leftarrow -L_{\text{signer}}$), which is not lower bounded, leading to high gradients and more difficult optimization. The latter addresses this problem by replacing this term with the absolute difference between the adversarial loss as defined in equation (4.4) and the signer classification loss (i.e. $L_{\text{adv}} \leftarrow |L_{\text{adv}} - L_{\text{signer}}|$). This option has a nice information theoretic interpretation as being an empirical upper-bound for the mutual information between the distribution of signer identities and the distribution of latent representations. Nonetheless, there exist infinitely many (non-uniform) distributions for which this loss vanishes. Our choice, besides being clearly lower bounded by the entropy of the uniform distribution, $\log k$, is minimum if and only if $p(\text{s} \mid g(\boldsymbol{X}_i; \boldsymbol{\theta}_g); \boldsymbol{\theta}_d) \equiv \mathcal{U}(\text{s})$, $\forall i$, meaning that the signer-classifier block is completely agnostic relatively to the signer identities of the training samples.

4.3.3.3 Signer-transfer training objective

To further encourage the latent representations $\boldsymbol{z}$ to be signer-invariant, we introduce an additional term in objective (4.5), the so-called signer-transfer loss L_{transfer}. The core idea of L_{transfer} is to match first order statistics of different signers earlier in the network. For this purpose, let $g^{(l)}(\cdot; \boldsymbol{\theta}_g)$ be the l-th layer of the encoder network, $l \in \{1, ..., m\}$, and consider the distance $\mathcal{D}^{(l)}(s, s'; \boldsymbol{\theta}_g)$ between two distinct signers s and s', defined as:

$$\mathcal{D}^{(l)}(s, s'; \boldsymbol{\theta}_g) \triangleq \left|\left| \frac{1}{n_s} \sum_{i=1}^{n} g^{(l)}(\boldsymbol{X}_i; \boldsymbol{\theta}_g) \mathbf{1}_{s_i = s} - \frac{1}{n_{s'}} \sum_{j=1}^{n} g^{(l)}(\boldsymbol{X}_j; \boldsymbol{\theta}_g) \mathbf{1}_{s_j = s'} \right|\right|^2, \tag{4.6}$$

n_s and $n_{s'}$ denote the number of training examples for signers s and s', respectively. Accordingly, the signer-transfer loss at the l-th layer is the sum of the pairwise distances between all signers, i.e.:

$$L^{(l)}_{\text{transfer}}(\boldsymbol{\theta}_g) \triangleq \sum_{\substack{s, s'=1 \\ s' \neq s}}^{k} \mathcal{D}^{(l)}(s, s'; \boldsymbol{\theta}_g). \tag{4.7}$$

The overall signer-transfer loss L_{transfer} is then a weighted sum of the losses computed at each layer of the encoder network:

$$L_{\text{transfer}}(\boldsymbol{\theta}_g) \triangleq \sum_{l=1}^{m} \beta^{(l)} L^{(l)}_{\text{transfer}}(\boldsymbol{\theta}_g), \tag{4.8}$$

where $\beta^{(l)} \geq 0$ is a hyperparameter that controls the relative importance of the loss obtained at the l-th layer. By combining (4.5) and (4.8), the encoder and sign-classifier networks are trained to minimize the following loss function:

$$\min_{\boldsymbol{\theta}_g, \boldsymbol{\theta}_h} \left\{ L(\boldsymbol{\theta}_g, \boldsymbol{\theta}_h, \boldsymbol{\theta}_d) \triangleq L_{\text{sign}}(\boldsymbol{\theta}_g, \boldsymbol{\theta}_h) + \mu_d L_{\text{adv}}(\boldsymbol{\theta}_g, \boldsymbol{\theta}_d) + \mu_t L_{\text{transfer}}(\boldsymbol{\theta}_g) \right\}, \tag{4.9}$$

where $\mu_t \geq 0$ is the weight that controls the relative importance of the signer-transfer term.

Summing up, the adversarial training procedure is organized by alternating between the minimization of objective (4.9) and the minimization of objective (4.1).

4.3.4 Experiments

4.3.4.1 Datasets

The experimental evaluation of the proposed model was performed using two publicly available SLR databases: the Jochen-Triesch database (Triesch and von der Malsburg [178]) and the Microsoft Kinect and Leap Motion American sign language (MKLM) database (Marin et al. [161, 162]).

Jochen-Triesch is a static hand posture database, which consists of 10 hand posture signs performed by a total of 24 subjects against three types of backgrounds: uniform light, uniform dark and complex (see Figure 4.3a). There exist three images for each subject and sign, one for each background type. The images of the Jochen-Triesch database are in grey-scale with a resolution of 128×128 pixels. Experiments on the Jochen-Triesch dataset were conducted using an available standard evaluation protocol for this dataset (Just et al. [179]), in which 6 subjects are used for training, 2 subjects are used for validation and hyperparameter tuning, and the remaining 16 are used for testing.

MKLM is a balanced dataset with 10 classes, representing 10 static gestures from the American sign language (see Figure 4.3b). Each sign was performed by 14 different people, and repeated 10 times, which results in a total of 1400 gestures. Contrary to the Jochen-Triesch database, there is no standard evaluation protocol for the MKLM database. To maximize the usage of the data in the evaluation process, the performance of the models was assessed using a five-fold cross-validation scheme with signer independence. This evaluation scheme yields at each split a training set composed by 10 signers, a validation set of 2 signers and a test set with the remaining 2 signers.

(A) Jochen-Triesch (B) MKLM

FIGURE 4.3: Illustrative samples of the two datasets used in the experiments.

In order to extract the manual signs from the noisy background of the images, the automatic hand detection algorithm (Ferreira et al. [180]) is used as a pre-processing step. The images are then cropped, resized to the average sign size of the training set, and normalized to be in the range $[-1, 1]$.

4.3.4.2 Baselines

Throughout this section, the proposed model is compared with state of the art methods for each dataset. Nevertheless, to further attest the robustness of the proposed model, two different baselines are also implemented:

- (Baseline 1) A CNN trained from scratch with weight decay regularization. For a fair comparison, the architecture of the baseline CNN corresponds to the architecture of the encoder network followed by the sign-classifier network of the proposed model.
- (Baseline 2) A CNN with the baseline 1 topology, but trained with the triplet loss (Schroff et al. [181]).

Here, the triplet loss concept is explored in order to impose signer-independence in the representation space and, hence, build up a more robust baseline. The underlying idea is to minimize the distance between an *anchor* and a *positive* latent representation, z_{y_i,s_i} and z_{y_p,s_p}, respectively; while maximizing the distance between the anchor z_{y_i,s_i} and a *negative* representation z_{y_n,s_n}. It is important to note that while anchor and positive latent representations have to be from the same sign class, their signer identity may or not change. On the other hand, anchor and negative representations are from different sign classes,

Hyperparameters	Symbol	Set
Leaning rate	–	$\{1e^{-04}, 1e^{-03}\}$
L^2-norm coefficient	–	$\{1e^{-05}, 1e^{-04}\}$
L_{triplet} weight	ρ	$\{0.1, 0.5, 1, 5, 10\}$
L_{adv} weight	μ_d	$\{0.1, 0.5, 0.8, 1, 3\}$
L_{transfer} weight	μ_t	$\{1.5e^{-04}, 2e^{-04}, 4e^{-04}, 1e^{-03}\}$

TABLE 4.1: Hyperparameter sets for the proposed adversarial model and baselines.

whereas their signer identity may also change. In order to train baseline 2 in an end-to-end fashion for sign classification, the overall loss function to be minimized is a trade-off between the triplet loss L_{triplet}, defined below, and the classification loss L_{sign}:

$$L_{\text{triplet}} \triangleq \frac{1}{n} \sum_{i=1}^{n} \left[||z_{y_i,s_i} - z_{y_p,s_p}||_2^2 - ||z_{y_i,s_i} - z_{y_n,s_n}||_2^2 + \alpha \right], \tag{4.10}$$

where $y_p = y_i$ and $y_n \neq y_i$ and the margin α enforced between *positive* and *negative* pairs was fixed at $\alpha = 1$. In addition, following Schroff et al. [181], we adopted an *online* triplet generation strategy, by selecting the hardest positive/negative samples within every mini-batch. The overall loss for this model is therefore $L_{\text{sign}} + \rho L_{\text{triplet}}$, where $\rho \geq 0$ is a hyperparameter.

All deep models were implemented in PyTorch and trained with the Adam optimization algorithm using a batch size of 32 samples. For reproducibility purposes, the source code as well as the weights of the trained models are publicly available online*. The hyperparameters that are common to all the implemented models (i.e. learning rate and L^2 regularization weight) as well as some hyperparameters that are specific to the proposed model (i.e. μ_d and μ_t) and to the implemented baseline 2 (i.e. ρ) were optimized by means of a grid search approach and cross-validation on the training set (see Table 4.1 for more details). The signer-transfer penalty L_{transfer} is applied to the last two layers of the encoder network with a relative weight of 1.

4.3.4.3 Results and discussion

Experiments on the Jochen-Triesch and MKLM databases are summarized in Table 4.2. We compare our method with other state of the art approaches that have published results on these datasets (Marin et al. [161], Kelly et al. [175], Dahmani and Larabi [176], Just et al. [179], Ferreira et al. [180]). The results on the Jochen-Triesch database are presented in terms of average classification accuracy in the overall test set as well as against each

*https://github.com/pmmf/SI-SLR

	Jochen Triesch			MKLM		
	Uniform Bg.	Complex Bg.	Both	Avg. ± std.	min	max
Just et al. [179]	92.8	81.3	87.9	–	–	–
Kelly et al. [175]	91.8	–	–	–	–	–
Dahmani and Larabi [176]	93.1	–	–	–	–	–
Marin et al. [161]	–	–	–	89.7	–	–
Ferreira et al. [180]	–	–	–	93.2	–	–
CNN (baseline 1)	97.50	74.4	89.8	89.9 ±8.8	73.0	98.0
CNN with triplet loss (baseline 2)	98.13	75.6	90.6	91.4 ±3.9	86.5	96.5
Proposed method	**98.8**	**91.3**	**96.3**	**94.8** ±3.5	**89.5**	**100.0**

TABLE 4.2: Classification accuracy (%) of the proposed adversarial method and baselines on Jochen-Triesch and MKLM datasets.

specific background type (i.e. uniform and complex). For the MKLM database, the table shows the average classification accuracy computed across all test splits, as well as the minimum and maximum accuracy value achieved by each method.

The most relevant observation is the superior performance of the proposed model. Specifically, this model exhibits the best overall classification accuracy on both SLR databases, clearly outperforming both implemented baselines and all the previous state of the art models. In Jochen-Triesch, the most challenging data are the images with complex background, where the proposed model surpasses all the remaining by a large margin. In addition, by analyzing the standard deviation as well as the minimum and maximum accuracy values, it is possible to observe that the proposed model is the method with the lowest variability, yielding consistently high accuracy rates across all test splits of the MKLM dataset. These results attest its robustness and capability to better deal with the large inter-signer variability that exists in the manual signing process of sign languages. Interestingly, the obtained results also reveal that the implemented baselines are in fact fairly strong models, both of them outperforming most of the state of the art methods on both datasets.

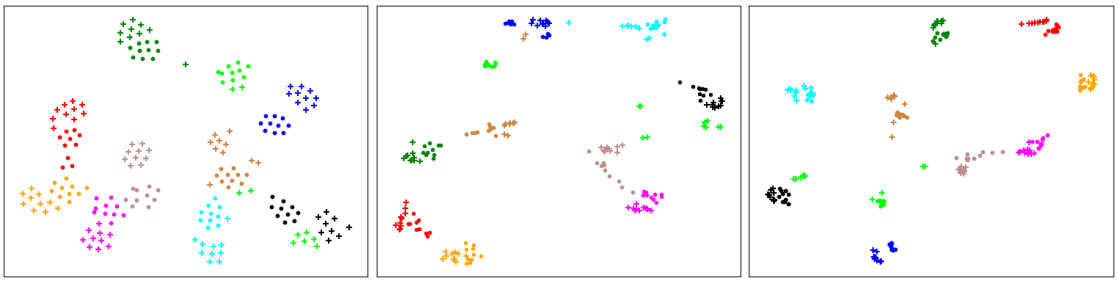
Table 4.3 illustrates the effect of each proposed training scheme by itself. For this purpose, the proposed model was trained either (i) with just the adversarial procedure, without the signer-transfer L_{transfer} loss, or (ii) with just the L_{transfer} penalty on the encoder network without adversarial training. The results clearly demonstrate the complementary effect between the two training procedures, as their combination provides the best overall classification accuracy. Interestingly, each training scheme outperforms on its own both baselines and state of the art methods.

	Adversarial (L_{adv}) only	Signer-transfer ($L_{transfer}$) only	Both
Jochen-Triesch	95.2	94.4	**96.3**
MKLM	94.0	94.1	**94.8**

TABLE 4.3: The effect of each training procedure in the proposed model. The results in the last column are replicated from Table 4.2 as they include both training procedures.

4.3.4.4 Latent space visualization

To further demonstrate the effectiveness of the proposed model in promoting signer-invariant latent representation spaces, we show in Figure 4.4 a visual inspection of the latent representations through the t-distributed stochastic neighbor embedding (t-SNE, van der Maaten and Hinton [182]). These plots clearly demonstrate the better capability of the proposed model of imposing signer-independence in the latent representations. The proposed model yields a latent representation space in which representations of different signers and same class are close to each other and well mixed, while it keeps latent representations of different classes far apart. By analyzing the t-SNE plot of baseline 1, it is possible to observe that the latent representations of different signers and the same class tend to be far apart in the latent space. In addition, there is some overlapping between clusters of different classes. Although baseline 2 (CNN with the triplet loss) promoted slightly improvements over the standard baseline CNN, the proposed model achieved by far the best signer-invariance and class separability.



(A) CNN – baseline 1 (B) CNN with triplet loss – baseline 2 (C) Proposed model

FIGURE 4.4: Two-dimensional projection of the latent representation space using the t-distributed stochastic neighbor embedding (t-SNE). Markers • and + represent 2 different test signers, while the different colors denote the 10 sign classes.

4.3.5 Conclusion

This paper presents a novel adversarial training objective, based on representation learning and deep neural networks, specifically designed to tackle the signer-independent SLR

problem. The underlying idea is to learn signer-invariant latent representations that preserve as much information as possible about the signs, while discarding the signer-specific traits that are irrelevant for sign recognition. For this purpose, we introduce an adversarial training procedure for simultaneously training an encoder and a sign-classifier over the target sign variables, while preventing the latent representations of the encoder to be predictive of the signer identities. To further discourage the underlying representations of retaining any signer-specific information, we propose an additional training objective that approximates the means of the latent distributions of different signers. Experimental results demonstrate the effectiveness of the proposed model in several SLR databases.

4.4 Adversarial domain generalization for iris presentation attack detection

In this section, we show how our adversarial domain generalization model for signer-invariant SLR (presented in Section 4.3) can be adapted to a specific biometrics-related application.

4.4.1 Introduction

Biometric recognition systems are considered reliable enough to be deployed in government and civilian applications. The shift from controlled samples acquisition to a more autonomous one increased the vulnerabilities of these systems. Unfortunately, presentation attack detection (PAD) measures had not grown robustly along with this quick evolution and several weak points can be exploited when performing unsupervised biometric identification as such in mobile biometrics, for example. Successful spoofing attempts have been made public in a matter of days, or even hours, after the release of high-tech devices equipped with biometric recognition. The iris recognition sensor of Samsung S8 was reportedly spoofed by German researchers by simply printing a photo of the authorised user and placing a contact lens in it [183]. More recently, the quick hack of Samsung Galaxy S10 ultrasonic fingerprint sensor suggests no presentation attack detection measures of any kind. It is fair to conclude that industry does not share the same enthusiasm as academic community on anti-spoofing measures denoted by the good amount of research continuously produced (Raghavendra and Busch [184], Czajka and Bowyer [185], Galbally et al. [186], Scherhag et al. [187]).

Fortunately, exceptions are starting to show in commercial products, like the recent case of the Apple iPhone "Face ID" case* or the FaceTec ZoOm® technology [188]. Undoubtedly this change is motivated and supported by initiatives that encourage the development and 'open testing' of spoofing coutermeasures such as 'The National Voluntary Laboratory Accreditation Program" (NVLAP) from NIST.†

Nevertheless, research-wise there are still open problems to address. Here, we focus on the fact that most PAD techniques are based on falsely optimistic evaluation methodologies (Sequeira et al. [189]): traditionally, the classification models are designed and then evaluated using datasets comprising *bona fide* presentations and a specific species of presentation attack instruments (PAI). The case when a PAI in the test set is significantly different from the ones used for training is overlooked. What if such sample has a higher probability to circumvent the system than the ones drawn from the original training dataset? To solve this research question it is necessary to develop robust methods to cope with sophisticated and unseen attacks as our eventual intruders become more capable and successfully develop new spoofing techniques.

The aforementioned problem has been addressed before regarding iris, fingerprint and face (often targeted under the open-set or anomaly detection contexts). However, it still remains a challenging topic. Despite the importance of iris as a biometric trait for recognition purposes, in our view, the study of iris PAD generalization problem to unseen PAI species (PAIS) has not been yet fully studied in literature.

The remainder of this section is organized as follows: i) we start by summarizing the related work on the topic (Section 4.4.2); ii) we formalize the problem and emphasize the necessary modifications that had to be done to the model presented in Section 4.3 to adapt it to this new application (Section 4.4.3); iii) we present experimental results that confirm the effectiveness of the model (Section 4.4.4); iv) we conclude this section with some final remarks (Section 4.4.5).

4.4.2 Related work

Recent PAD methods in general, and iris-focused ones in particular, have demonstrated remarkable performances. However, a methodological limitation can be pointed as it is

*www.biometricupdate.com/201812/android-devices-facial-recognition-fooled-by-3d-printed-head-but-not-face-id

†The NVLAP provides third-party accreditation to testing and calibration laboratories in response to legislative actions or requests from government agencies or private-sector organizations. NVLAP-accredited laboratories are assessed against the management and technical requirements from ISO/IEC 17025:2017.

recurrently found that these results are obtained when training and test data comprise the same type of attacks, i.e. the same PAIS. This problem has been addressed and proved that the performance rates of these PAD methods typically decrease significantly when the PAIS is new to the system (Sequeira et al. [189], Marasco and Sansone [190], Bowyer and Doyle [191]). This performance drop can result from the large inter-PAIS variability. A practical PAD system must operate in a PAIS-independent scenario, which means that the type of PAIS of the test set must not be seen during the training routine of the models. This problem is one of the crucial problems for the development of real-world PAD systems and it has frequently been tackled in literature as an open-set or anomaly detection problem.

The pioneer work that raised the evaluation of PAD methods across different types and unseen PAIS appeared in the fingerprint domain with the work of Marasco and Sansone [190]. Rattani et al. [192] and Sequeira and Cardoso [193], despite using different approaches, both relied on the idea of enforcing the knowledge of the bona fide presentations over the attacks to better deal with unseen PAIS. Bowyer and Doyle [191] studied the evaluation of a binary classification on contact lenses iris spoofing attacks. By using an unseen type on the test set the authors showed that using the same lens types in both the training and testing data can give a very misleading idea of the accuracy of the method.

A step forward was made by combining methodologies designed for print and contact lenses attack (Sequeira et al. [194]). Eventually, the construction of a new database comprising several types of iris PAIS (Raghavendra and Busch [184]) allowed new evaluation scenarios. Sequeira et al. [189] showed that whenever a new PAIS is presented in the test step, the performance of the classifier drops significantly and that an improvement can be obtained when a one-class classifier is trained only with bona fide presentations.

One-class classification was also used for face by Arashloo et al. [195]. With the rise of deep learning (DL) techniques, PAD methods have been proposed applying deep representations for iris, face and fingerprint (Menotti et al. [196], Pinto et al. [197]), following the same binary approach. Recent works investigate the robustness of DL fingerprint PAD methods to deal with unseen PAI species (Tolosana et al. [198]).

Until recently, most of the proposed approaches either make overly optimistic assumptions about the attacker (binary classification approaches) or only use part of the data (and therefore, of the knowledge) available at training time to design the models (one-class approaches). Therefore, the goal of this work is to present an iris PAD method

that uses the information of both bona fide and available attack presentations and is robust to unseen PAI species. This goal will be achieved by enforcing the learning of the task of distinguishing the bona fide from the attack presentations while at the same time ensuring the invariance between the different type of the PAI species.

4.4.3 Methodology

The approach adopted here coincides in most aspects with the one described in Section 4.3, so we shall focus on describing the slight differences that exist. Now, the data consists of $\{\boldsymbol{X}_i^{(\mathrm{bf})}\}_{i=1}^{n_{\mathrm{bf}}} \cup \{(\boldsymbol{X}_i^{(\mathrm{a})}, s_i)\}_{i=1}^{n_{\mathrm{a}}}$, i.e. there is one set containing n_{bf} bona fide examples and another one containing n_{a} attack examples. Each attack example $\boldsymbol{X}_i^{(\mathrm{a})}$ is annotated with the corresponding PAI species label $s_i \in \{1, 2, \ldots, k\}$.

In this problem, we are solely interested in classifying samples as bona fide or attack, thus $h(\cdot; \boldsymbol{\theta}_h)$ (formerly designated as sign-classifier) is now a binary classifier. More importantly, we want to obtain latent representations that are invariant to the PAI species, but the latent representations of bona fide examples should be easily separable from these. Thus, now, only the attack samples are fed through adversarial classifier $d(\cdot; \boldsymbol{\theta}_d)$ (formerly designated as signer-classifier) and used for the adversarial training routine. For the same reason, the transfer loss L_{transfer} approximating first-order statistics only applies to these samples too. Figure 4.5 presents the model architecture and hopefully makes the differences between this and our previous model even clearer.

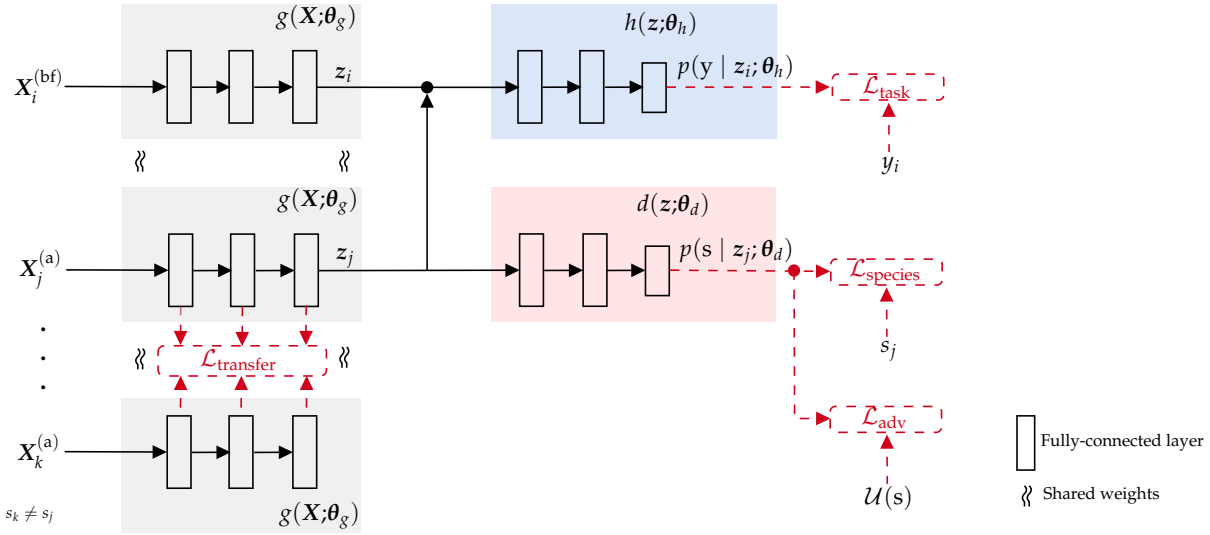


FIGURE 4.5: Block diagram of the proposed species-invariant neural network.

	Attack i)	Attack ii)	Attack iii)	Attack iv)	Attack v)	Avg.
wLBP+SVM [189]	78.9	90.4	**98.1**	**95.7**	97.1	92.0
Baseline wLBP+MLP	78.0	**93.0**	94.5	90.0	95.5	90.2
Proposed wLBP+MLP$_{adv}$	**82.0**	**93.0**	98.0	94.5	**97.5**	**93.0**

TABLE 4.4: Presentation attack detection accuracy (%) in the VSIA dataset.

4.4.4 Experiments

We use the Visible Spectrum Iris Artefact (VSIA) database (Raghavendra and Busch [184]) in our experiments. This dataset comprises five different presentations combining print and electronic screen attacks: i) Print Attack (PA); ii) iPad Electronic Screen Attack (ESA); iii) Samsung Galaxy Tab ESA; iv) combined PA & ESA using iPad; and v) combined PA & ESA using Samsung Pad. The methods are evaluated by leaving out one PAI species for testing. The development set is therefore divided into one species for validation and the remaining for training. Also the same set of samples are used for testing across the different experiments to allow precise comparison of the results. Following Sequeira et al. [189], weighted local binary pattern features (wLBP, Zhang et al. [199]) were extracted in a preprocessing step and fed as input to the network, which in this case consists of an MLP.

Our model was compared to a baseline consisting of the same classifier without the PAI species classifier and adversarial training and to an SVM operating on top of the same wLBP features (Sequeira et al. [189]). Results are in Table 4.4. Comparing the accuracy for each attack, it can be observed that uniquely replacing the SVM with an MLP does not result in an improvement. This can be explained by the fact that the dataset has a very limited size and therefore the MLP method tends to overfit due to the lack of training samples. It was not for no reason that SVMs ruled for a long time in the pattern recognition domain. However, the proposed adversarial approach outperformed the SVM for most attacks and on average as well.

For further results and details about the experiments, please see Ferreira et al. [131].

4.4.5 Conclusion

This work proposed a method to improve the robustness and generalization capacity of an iris PAD method to new attacks. The goal of the proposed model is to learn latent representations invariant to the PAI species that preserve relevant information about the PAD properties while discarding the 'PAI-species'-specific aspects that may hamper the PAD classification task. The proposed regularization strategies made the PAD method

'PAI-species'-independent and robust to new test PAIS. The experiments were based in comparing a baseline MLP and an MLP trained with adversarial strategies using as input highly discriminative features (wLBP) extracted from the images. When comparing the baseline MLP to an SVM classifier the results are quite similar or even worse. This can be explained simply by the fact that the dataset has a very limited size and the MLP method will overfit. However, applying the adversarial regularization strategy significantly improved the PAD robustness of the method. The obtained results clearly suggest that the application of deep learning techniques with additional strategies will provide breakthroughs in this challenge.

4.5 DeSIRe: Deep Signer-Invariant Representations for Sign Language Recognition

4.5.1 Introduction

In Section 4.3, we introduced a method for domain generalization which uses adversarial neural networks to align the marginal distributions of multiple source domains. The method showed promising results for both visual (Section 4.3) and non-visual data (Section 4.4). Here, we again focus our attention on vision problems and, specifically, on solving the problem of signer-independent SLR.

To specifically tackle the signer-independent SLR problem, we now present DeSIRe, a novel deep neural network that aims to learn Deep Signer-Invariant Representations. The underlying idea is to explicitly enforce the model to automatically learn highly discriminative signer-invariant feature representations from the data by aligning and regularizing conditional distributions in a latent space. To accomplish this goal, the DeSIRe model consists of two main modules or components, namely a conditional variational autoencoder (CVAE) and a classifier. Specifically, the main task of the CVAE is to explicitly impose signer independence on the learned latent representations. This is achieved by encouraging the CVAE to learn latent representations whose conditional posterior distribution (given the image and the corresponding sign class label) is independent of the signer identity. Accordingly, the learned latent representations will preserve as much information as possible about the class (sign) and discard the irrelevant parts that are signer-specific. In addition, the CVAE acts as a teacher model for the classifier since the distribution over latent representations is used to regularize the hidden representations of the classifier.

These hidden representations are then fed into a multilayer perceptron (MLP) for sign classification. The result is a signer-independent model robust to new test signers.

The remainder of this section is organized as follows: i) the proposed signer-independent deep neural network along with the proposed loss function and regularization schemes are fully described in Section 4.5.2; ii) Section 4.5.3 reports the experimental evaluation of the proposed methodology, in which a comparison with state of the art and baseline methods is performed; iii) finally, conclusions and some topics for future work are presented in Section 4.5.4.

4.5.2 The DeSIRe model

The high-level block diagram of the proposed DeSIRe model is depicted in Figure 4.2. As it is possible to observe, it is composed by a CVAE and a classifier. In our model, the underlying idea of the CVAE is to learn an invertible mapping to a space where the signer-specific information is disentangled from the discriminative properties of the sign class. The CVAE can be thought as a teacher model for the classifier, as the distribution over latent representations $\mathbf{z}$ is used to regularize the hidden representations $\tilde{z}$ of the classifier. These hidden representations $\tilde{z}$ are then fed into a multilayer perceptron (MLP) for a robust signer-independent SLR.

Specifically, the CVAE consists of an encoder and a decoder network, parameterized by $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_d$, respectively. The purpose of the encoder network is to learn a distribution $q(\mathbf{z} \mid \mathbf{X}, \mathrm{y}, \mathrm{s}; \boldsymbol{\theta}_e)$ which approximates the true posterior distribution of the latent code $\mathbf{z}$ given the image $\mathbf{X}$, the class label y and the signer identity s. By conditioning the posterior distribution on s and y, we are empowering the encoder by learning a domain and class-dependent transformation. Here, the key idea is to learn latent codes whose conditional posterior distribution is independent of the signer identity, that is $q(\mathbf{z} \mid \mathbf{X}, \mathrm{y}, \mathrm{s}; \boldsymbol{\theta}_e) \equiv q(\mathbf{z} \mid \mathbf{X}, \mathrm{y}; \boldsymbol{\theta}_e)$. Equivalently, latent codes are conditionally independent of the signer identity given the image and its class if and only if:

$$q(\mathbf{z} \mid \mathbf{X}, \mathrm{y}, \mathrm{s} = s; \boldsymbol{\theta}_e) \equiv q(\mathbf{z} \mid \mathbf{X}, \mathrm{y}, \mathrm{s} = s'; \boldsymbol{\theta}_e), \tag{4.11}$$

for any two distinct signers s and s'. In order to promote this signer-independence property, the loss function includes a term that penalizes deviations from this equality. However, if no additional care is taken, this condition would compete with the reconstruction objective since reconstructing an image implies preserving as much information about the

image as possible, including signer-specific information. Therefore, the signer identity is sampled uniformly at random and fed as an additional input to the decoder network. By this mean, the decoder shall provide a disentangled representation of the signer identity which, combined with the signer-invariant latent code $\boldsymbol{z}$, will be used to reconstruct the original sample.

Intuitively, as the latent vector $\boldsymbol{z}$ is sampled from $q(\mathbf{z} \mid \mathbf{X}, \mathrm{y}, \mathrm{s}; \boldsymbol{\theta}_e)$, the latent representations will preserve as much information as possible about the sign class and discard the irrelevant parts that are characteristic of each signer. The loss function is defined to promote similarity between the latent codes $\boldsymbol{z}$ and the hidden representations $\tilde{\boldsymbol{z}}$ of the classifier module. The classifier is then trained on these signer-invariant representations for a robust signer-independent SLR. Formally, the classifier correspond to the function composition $h \circ g$, where $h(\cdot; \boldsymbol{\theta}_h) : \mathcal{Z} \mapsto \mathcal{Y}$ represents our task-specific function, parameterized by $\boldsymbol{\theta}_h$, that maps from the hidden representation to the predicted sign class $\hat{y}$, and $g(\cdot; \boldsymbol{\theta}_g) : \mathcal{X} \mapsto \mathcal{Z}$ denotes an encoding function, parameterized by $\boldsymbol{\theta}_g$, that maps input images to the corresponding hidden representations.

4.5.2.1 Loss function

The DeSIRe model trained to minimize the following loss function with respect to parameters $\Theta = \{\boldsymbol{\theta}_e, \boldsymbol{\theta}_d, \boldsymbol{\theta}_g, \boldsymbol{\theta}_h\}$:

$$L(\Theta) \triangleq L_{\text{CVAE}}(\boldsymbol{\theta}_d, \boldsymbol{\theta}_e) + \lambda_1 L_{\text{emb}}(\boldsymbol{\theta}_e, \boldsymbol{\theta}_f) + \lambda_2 L_{\text{class}}(\boldsymbol{\theta}_f, \boldsymbol{\theta}_g), \tag{4.12}$$

where $\lambda_1, \lambda_2 \geq 0$ are the weights that control the interaction between the loss terms.

The ultimate goal of the CVAE loss, L_{CVAE}, is to explicitly impose signer independence by learning latent representations which are conditionally independent from the signer identity. In this regard, L_{CVAE} is defined as:

$$L_{\text{CVAE}}(\boldsymbol{\theta}_d, \boldsymbol{\theta}_e) \triangleq L_{\text{rec}}(\boldsymbol{\theta}_d) + \alpha_1 L_{\text{prior}}(\boldsymbol{\theta}_e) + \alpha_2 L_{\text{signer\_inv}}(\boldsymbol{\theta}_e), \tag{4.13}$$

where $\alpha_1, \alpha_2 \geq 0$ are hyperparameters that control the relative importance of each loss term. The first two terms, L_{rec} and L_{prior}, correspond to the loss function of a standard CVAE, containing some special modifications for promoting signer-independence in the latent space. The reconstruction loss L_{rec} encourages the decoder to learn how to reconstruct the input data $\mathbf{X}$. For the decoder, we assume that the conditional likelihood of the data $\mathbf{X}$ given the latent code $\mathbf{z}$ and the signer identity s follows a Gaussian distribution.

Accordingly, as explained in Section **??**, the reconstruction loss corresponds to the mean-squared error between a training image and a generated image. Here, however, instead of working with pairs of ground-truth images together with their respective reconstructions, we make a slight modification that further promotes signer-invariant encodings. Let $\boldsymbol{X}_{y,s}^{(r)}$ denote the r-th image of signer s and sign class y. Specifically, we compute the mean-squared error between the j-th d-dimensional training image $\boldsymbol{X}_{y_j,s_j}^{(r_j)}$ and the generated d-dimensional image $\boldsymbol{\mu}_d(\boldsymbol{z}_i, s_j; \boldsymbol{\theta}_d)$ which is produced by the decoder when fed with the encoding $\boldsymbol{z}_i$ of the i-th training image $\boldsymbol{X}_{y_i,s_i}^{(r_i)}$ and with the signer identity s_j of the j-th training image:

$$L_{\text{rec}}(\boldsymbol{\theta}_d) \triangleq \frac{1}{nd} \sum_{i=1}^{n} ||\boldsymbol{X}_{y_j,s_j}^{(r_j)} - \boldsymbol{\mu}_d(\boldsymbol{z}_i, s_j; \boldsymbol{\theta}_d)||_2^2, \tag{4.14}$$

where $\boldsymbol{z}_i$ is sampled from $q(\mathbf{z}_i \mid \boldsymbol{X}_{y_i,s_i}^{(r_i)}, y_i, s_i; \boldsymbol{\theta}_e)$ using the reparameterization trick (**??**), and j is such that $y_j = y_i$, s_j is sampled from a distribution $w(\mathrm{s} \mid s_i)$, defined below, and r_j is sampled uniformly from the set of available repetitions:

$$w(\mathrm{s} \mid s_i) \triangleq \begin{cases} 1 - \rho, & \mathrm{s} = s_i, \\ \frac{\rho}{k-1}, & \mathrm{s} \in \{1, 2, \ldots, k\} \setminus \{s_i\}. \end{cases} \tag{4.15}$$

Here, as before, $\{1, 2, \ldots, k\}$ is the set of signer identities in the training data and $\rho \in [0, 1]$ is a hyperparameter. By sampling the identity s_j of the ground-truth image from $w(\mathrm{s} \mid s_i)$, the decoder will be trained to reconstruct an image of a different subject (but same sign class) than the one that was used to produce the encoding. This will happen in a proportion ρ of the cases. This procedure further disentangles signer-specific information from the sign class and therefore aims to reduce inter-signer variability. On the other hand, by sampling the sign repetition r_j, the decoder will also be trained to reconstruct a distinct image of the same person and sign class as the image that produced the encoding. Here, the purpose is to gain robustness to intra-signer variability. Although less problematic than the former, this type of variability is also relevant since the same signer does not always repeat the same sign in exactly the same way. Moreover, different image acquisition conditions (e.g. background, illumination, distance to the camera, etc.) from one repetition to another also result in intra-signer variability.

The L_{prior} term corresponds to the KL divergence between the posterior and the prior as commonly used in a standard CVAE:

$$\begin{aligned} L_{\text{prior}}(\boldsymbol{\theta}_e) &\triangleq \frac{1}{nl}\sum_{i=1}^{n} D_{\text{KL}}(q(\mathbf{z}_i \mid \boldsymbol{X}_{y_i,s_i}^{(r_i)}, y_i, s_i; \boldsymbol{\theta}_e)||\mathcal{N}(\mathbf{z}_i; 0, \boldsymbol{I})) \\ &= \frac{1}{2nl}\sum_{i=1}^{n}\sum_{j=1}^{l}\left(\mu_j^{(i)^2} + \sigma_j^{(i)^2} - 1 - 2\log\sigma_j^{(i)}\right), \end{aligned} \tag{4.16}$$

where l is the dimension of the latent space and $\mu_j^{(i)}$ and $\sigma_j^{(i)}$ denote the j-th elements of the vectors $\boldsymbol{\mu}_e(\boldsymbol{X}_i, y_i, s_i; \boldsymbol{\theta}_e)$ and $\boldsymbol{\sigma}_e(\boldsymbol{X}_i, y_i, s_i; \boldsymbol{\theta}_e)$, respectively.

An explicit constraint for signer-independence is also introduced in the CVAE loss function. $L_{\text{signer\_inv}}$ encourages the conditional posterior distribution of latent codes $\mathbf{z}$, given the image $\mathbf{X}$ and its class y, to be independent of the signer identity s. This loss is defined as the KL divergence between conditional posterior distributions of $\mathbf{z}$, conditioned on the same class but also on different signer identities:

$$\begin{aligned} \mathcal{L}_{\text{signer\_inv}}(\boldsymbol{\theta}_e) &\triangleq \frac{1}{nl}\sum_{i=1}^{n} D_{\text{KL}}\left(q(\mathbf{z}_i \mid \boldsymbol{X}_{y_i,s_i}^{(r_i)}, y_i, s_i; \boldsymbol{\theta}_e)\middle|\middle|q(\mathbf{z}_k \mid \boldsymbol{X}_{y_k,s_k}^{(r_k)}, y_k, s_k; \boldsymbol{\theta}_e)\right) \\ &= \frac{1}{2nl}\sum_{i=1}^{n}\sum_{j=1}^{l}\left(\frac{(\mu_j^{(i)} - \mu_j^{(u)})^2}{\sigma_{e,k,j}^2} + \frac{\sigma_j^{(i)^2}}{\sigma_j^{(u)^2}} - 1 + 2\log\sigma_j^{(u)} - 2\log\sigma_j^{(i)}\right), \end{aligned} \tag{4.17}$$

where u is such that $y_u = y_i$ and s_u takes values in $\{1, 2, \ldots, k\} \setminus \{s_i\}$ with equal probability. The second equality follows from the fact that both distributions are Gaussian and so their KL divergence may be computed analytically, as in equation (4.16).

The signer-invariant latent representations $\mathbf{z}$ learned by the CVAE are then used to regularize the hidden representations $\tilde{\boldsymbol{z}}$ of the classifier. Such regularization is promoted by the L_{emb} loss term, which encourages the latent representations of the CVAE and the classifier to be as similar as possible. Following this idea, the embedding loss L_{emb} is defined as the expected mean-squared error between $\mathbf{z}$ and $\tilde{\boldsymbol{z}}$, that is:

$$L_{\text{emb}}(\boldsymbol{\theta}_e, \boldsymbol{\theta}_g) \triangleq \frac{1}{nl}\sum_{i=1}^{n} \mathbb{E}_{\mathbf{z}_i \sim q(\mathbf{z}|\boldsymbol{X}_{y_i,s_i}^{(r_i)}, y_i, s_i; \boldsymbol{\theta}_e)}\left[||\mathbf{z}_i - \tilde{\boldsymbol{z}}_i||^2\right]. \tag{4.18}$$

In practice, we replace equation (4.18) by its empirical approximation with one sample, which yields:

$$L_{\text{emb}}(\boldsymbol{\theta}_e, \boldsymbol{\theta}_g) \approx \frac{1}{nl}\sum_{i=1}^{n} ||\boldsymbol{z}_i - \tilde{\boldsymbol{z}}_i||^2, \tag{4.19}$$

where $\boldsymbol{z}_i$ is sampled from $q(\mathbf{z}_i \mid \boldsymbol{X}_{y_i,s_i}^{(r_i)}, y_i, s_i; \boldsymbol{\theta}_e)$, again using the reparameterization trick (**??**). This approximation has an extra regularizing effect on the classifier network, by

introducing some stochastic noise in its training routine.

Finally, the classification loss, L_{class}, trains the model to predict the output sign labels and corresponds to the categorical cross-entropy, defined by:

$$L_{\text{class}}(\boldsymbol{\theta}_g, \boldsymbol{\theta}_h) \triangleq -\frac{1}{n}\sum_{i=1}^{n} \log p(y_i \mid \boldsymbol{X}_{y_i,s_i}^{(r_i)}; \boldsymbol{\theta}_g, \boldsymbol{\theta}_h), \tag{4.20}$$

where $p(y \mid \boldsymbol{X}; \boldsymbol{\theta}_g, \boldsymbol{\theta}_h)$ is the predicted probability that a given image $\boldsymbol{X}$ belongs to its ground-truth class y, according to the current classifier parameters $\boldsymbol{\theta}_g$ and $\boldsymbol{\theta}_h$.

A full schematic of the DeSIRe model including all loss terms is presented in Appendix C.1. A few training heuristics which were employed to improve the training of the model are described in Appendix C.2.

4.5.2.2 Inference

During the training stage, the CVAE module plays the role of a teacher model for the classifier. Hence, the CVAE can be discarded at inference time. Therefore, inference in DeSIRe simply consists of a forward pass through the classifier network: $\tilde{\boldsymbol{z}} = g(\boldsymbol{X}; \boldsymbol{\theta}_g)$ and $\hat{y} = h(\tilde{\boldsymbol{z}}; \boldsymbol{\theta}_h)$.

4.5.3 Experiments

For reproducibility purposes, the source code as well as the weights of the trained models are publicly available online*. Further implementation detailes are described in Appendix C.3.

4.5.3.1 Datasets

We use the same datasets and follow the same experimental protocol as in our previous model, described in Section 4.3.4. Additionally, we also evaluate our model in a subset of the CorSiL database.

CorSiL is a dataset for Portuguese sign language and expressiveness recognition. Its SLR subset comprises 182 isolated signs and 40 continuous sequences, which we have further refined by selecting 31 isolated signs from 11 distinct signers, with each sign being repeated 3 times for each signer. All signs were performed in a free and natural signing environment. This variability, together with the large number of sign classes, makes this

*https://github.com/pmmf/DeSIRe

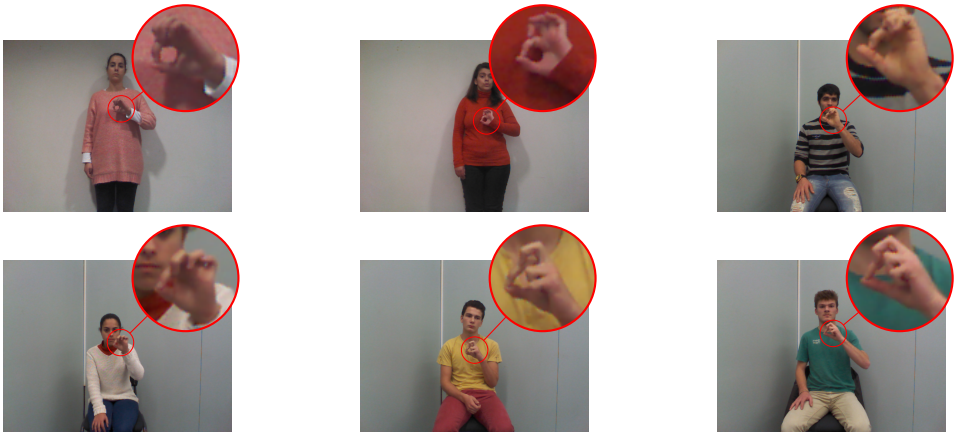


FIGURE 4.6: Illustration of the inter-signer variability using some samples of the CorSiL database. The six signers are performing the sign "eight" of the Portuguese sign language.

dataset a challenging one. A few samples are shown in Figure 4.6. We use six signers for training, one signer for validation and the remaining four signers are used for testing.

4.5.3.2 Baselines

In addition to the baselines used in Section 4.3.4 and to our previous model, we implemented two further models for comparison here:

- DANN, by Ganin and Lempitsky [69], which we have already discussed extensively. The application of this method to our problem implied two main changes in the original method: i) the binary domain-classifier (source vs. target domain) was extended to k classes (number of signers in the training data); ii) since our data is fully annotated (sign classes and signer identities are always available), training was performed in a fully supervised fashion.

- DTML, by Hu et al. [200], a reconstruction-based domain adaption algorithm. The implementation of this methodology for our particular task also implied the generalization of the original model from one single domain to k source domains.

4.5.3.3 Results and discussion

Table 4.5 shows the classification accuracy of the proposed DeSIRe model and the two aforementioned baselines on Jochen-Triesch and MKLM datasets. For easy comparison, we replicate the results from Table 4.2 for our adversarial model (Section 4.3). The results

	Jochen Triesch			MKLM		
	Uniform Bg.	Complex Bg.	Both	Avg. ± std.	min	max
DANN [69]	98.1	83.8	93.3	94.3 ±2.5	91.5	96.5
DTML [200]	98.8	85.6	94.4	94.1 ±3.8	87.0	97.5
Proposed adv. model (Section 4.3)	98.8	91.3	96.3	94.8 ±3.5	89.5	**100.0**
DeSIRe	**99.7**	**92.5**	**97.3**	**96.8** ±2.4	**93.0**	99.0

TABLE 4.5: Classification accuracy (%) of DeSIRe and baselines on Jochen-Triesch and MKLM datasets.

for the remaining baselines are omitted since they are the same as before and inferior to those of our adversarial model.

A first observation is the superior performance of DeSIRe. It is also worth mentioning that our previous model for domain generalization outperformed DANN and DTML in most settings. Nevertheless, DeSIRe achieved the best overall classification accuracy in all settings. Another interesting observation is the performance of DeSIRe against complex backgrounds, which even exceeds that of our previous model. These results demonstrate the robustness of the proposed model to inter-signer variability as well as its capability of dealing with the large intra-signer variability of this dataset. As previously explained in Section 4.5.2.1, the robustness to intra-signer variability is mostly due to the proposed sampling scheme of the sign repetition introduced to the decoder network, which enforces the learned latent representations to discard this type of variability. Another important reason that may explain the superiority of DeSIRe is the fact that this model aligns class-conditional distributions, rather than marginal ones. Since all signers have labeled data and, in the datasets we use here, the marginal class distributions are balanced across signers, aligning marginal feature distributions is less problematic here than it was for domain adaptation (Chapter 3). Anyway, the explicit alignment of conditional distributions is preferable since it promotes class clustering in the latent space. Additionally, as the encoder is conditioned on the signer identity, the feature transformation learned by DeSIRe is domain-dependent, which is not the case for the remaining models.

The classification accuracy for sign classification in the CorSiL database is presented in Table 4.6. As the SI-PSL database contains a large number of sign classes (31), the results are presented in terms of top-1, top-3 and top-5 classification accuracy. As shown in the table, DeSIRe outperformed both the implemented baselines and the state of the art domain adaption methods in all the three classification metrics. However, it should be noticed that, regardless of the employed methodology, the overall performance in this database is significantly below that obtained in the other two databases. These results

	Top-1	Top-3	Top-5
DANN [69]	48.7	75.5	83.3
DTML [200]	39.3	68.0	79.8
CNN (baseline 1)	46.0	74.7	86.8
CNN with triplet loss (baseline 2)	42.7	72.3	82.0
Proposed adv. model (Section 4.3)	49.1	76.0	85.2
DeSIRe	**51.9**	**76.6**	**87.9**

TABLE 4.6: Classification accuracy (%) of DeSIRe and baselines on the CorSiL dataset.

attest to the difficulty of the classification task on the presented database and should encourage further research on signer-independent SLR.

Since DeSIRe is a rather complex model, with multiple loss terms and techniques to enhance its performance, we believe it is highly instructive to provide a more thorough analysis of its training behavior and further experimental results. Both are presented in Appendix C. Specifically, we present a qualitative and quantitative analysis of the learned latent representations in Appendices C.4 and C.5, respectively. We illustrate the interaction of the multiple loss terms in Appendix C.6. A hyperparameter sensitivity analysis is conducted in Appendix C.7.

4.5.4 Conclusion

The work we have just presented addresses the topic of signer-independent sign language recognition. To tackle this problem, we proposed a novel end-to-end deep neural network along with a well-designed loss function that explicitly models signer-invariant latent representations from the data. Specifically, the proposed model is composed by two main modules: i) a conditional variational autoencoder (CVAE) and ii) a classifier. The purpose of the CVAE module is to learn latent representations of the input data whose conditional posterior distribution, given the image and its sign label, is independent of the signer identity. During the training stage, the CVAE plays the role of a teacher model for the classifier, since the conditional posterior distribution over latent representations is used to regularize the hidden representations of the classifier. These signer-invariant hidden representations are then used for a robust signer-independent SLR recognition. Experimental results demonstrate the robustness of the proposed model to new test signers. The proposed model provides quite promising results, outperforming the implemented

baseline methods and current state-of-the-art SLR and domain adaptation methods in different SLR databases. Extending the proposed DeSIRe model to dynamic signs (i.e. video) is an interesting direction for future work.

4.6 Summary

In this chapter, we treated the problem of domain generalization. As we have seen, DG is closely related to domain adaptation, which was our focus in Chapter 3. The main difference is the fact that in DG the target domain is not known beforehand, i.e. at training time, so no data from such domain is available, neither labeled nor unlabeled. The goal is then to build a model that can generalize well to new target domains, which are assumed to be fairly similar to the source ones. In practice, we have seen that most algorithms for DA can be adapted to the problem of DG with minor modifications.

Sign language recognition was the motivating application for both methods presented in this chapter. This is a problem where the DG setting fits perfectly, as an SLR automatic system should ideally perform consistently well regardless of the signer identity. For this purpose, we first presented a model in Section 4.3 that builds upon existing adversarial-based methods for DA and DG. The main difference is a novel definition for the adversarial objective. Instead of maximizing the discriminator loss, our adversarial routine aims to force this module to output a uniform distribution over the signer identities. This results in a smoother objective with bounded gradients, besides providing a more interpretable way to assess the signer-independence of the learned representations. In Section 4.4, we showed that a very similar idea can be applied to increase the robustness of iris presentation attack detection methods to unseen attacks. Since the input data used there were not images, this experiment validated the effectiveness of the proposed adversarial model for non-visual data as well.

In Section 4.5, we introduced another solution for the DG problem in the context of SLR, which outperforms the former and all baselines we have compared it with. Our DeSIRe model uses a variational autoencoder to learn signer-invariant representations which are used to train a non-probabilistic feature extractor. Apart from the multiple training strategies that enhance the robustness of the learned representations, the superior performance of DeSIRe is likely explained by the fact that it aligns class conditional distributions, instead of marginals, and also by the domain-dependent feature transformation learned by the encoder network.

Appendix A

SpaMHMM – supplementary material

A.1 Derivation of the EM learning algorithms for MHMM and SpaMHMM

A.1.1 EM for MHMM (Algorithm 2.3)

Algorithm 2.3 follows straightforwardly from applying EM to the model defined by equations (2.18) and (2.19) with the objective (2.20). As mentioned in Section 1.4.4, for table CPDs, the E-step consists in finding expected sufficient statistics $M(w, v)$ and $M(v)$, as defined in equations (1.17) and (1.18), for each $(w, v) \in \mathrm{Val}(\mathrm{w}, \mathrm{Pa}_{\mathcal{G}}(\mathrm{w}))$ and for each w in $\mathcal{G}$. In this case, $\mathcal{G}$ is the Bayesian network in Figure 2.5. Thus, in our setting, those equations yield:

$$M(z, y) = \sum_{i=1}^{n} p(y, z \mid \boldsymbol{x}_i, y_i) = \sum_{i=1}^{n} p(z \mid \boldsymbol{x}_i, y_i) \mathbf{1}_{y_i = y}, \tag{A.1}$$

$$M(y) = \sum_{i=1}^{n} p(y \mid \boldsymbol{x}_i, y_i) = \sum_{i=1}^{n} \mathbf{1}_{y_i = y}, \tag{A.2}$$

$$\begin{aligned} M(\mathrm{h}^{(0)} = h, z) &= \sum_{i=1}^{n} p(\mathrm{h}^{(0)} = h, z \mid \boldsymbol{x}_i, y_i) = \sum_{i=1}^{n} p(\mathrm{h}^{(0)} = h \mid \boldsymbol{x}_i, y_i, z) p(z \mid \boldsymbol{x}_i, y_i) \\ &= \sum_{i=1}^{n} p(\mathrm{h}^{(0)} = h \mid \boldsymbol{x}_i, y_i) p(z \mid \boldsymbol{x}_i, y_i), \end{aligned} \tag{A.3}$$

$$M(z) = \sum_{i=1}^{n} p(z \mid \boldsymbol{x}_i, y_i), \tag{A.4}$$

$$M(\mathrm{h}^{(t)} = h', \mathrm{h}^{(t-1)} = h, z) = \sum_{i=1}^{n} p(\mathrm{h}^{(t)} = h', \mathrm{h}^{(t-1)} = h, z \mid \boldsymbol{x}_i, y_i)$$

$$= \sum_{i=1}^{n} p(\mathrm{h}^{(t)} = h' \mid \mathrm{h}^{(t-1)} = h, \boldsymbol{x}_i, z) p(\mathrm{h}^{(t-1)} = h \mid \boldsymbol{x}_i, z) p(z \mid \boldsymbol{x}_i, y_i), \quad \text{(A.5)}$$

$$M(\mathrm{h}^{(t-1)} = h, z) = \sum_{i=1}^{n} p(\mathrm{h}^{(t-1)} = h, z \mid \boldsymbol{x}_i, y_i)$$

$$= \sum_{i=1}^{n} p(\mathrm{h}^{(t-1)} = h \mid \boldsymbol{x}_i, z) p(z \mid \boldsymbol{x}_i, y_i). \quad \text{(A.6)}$$

These equations allow us to compute the following updates in the M-step:

$$\alpha_z^{(y)} = \frac{M(y, z)}{M(y)} \quad \text{and} \quad \pi_h^{(z)} = \frac{M(\mathrm{h}^{(0)} = h, z)}{M(z)}. \quad \text{(A.7)}$$

For the state transition matrices update, the same idea applies after summing over the sequence length:

$$A_{h,h'}^{(z)} = \frac{\sum_{t=1}^{T} M(\mathrm{h}^{(t)} = h', \mathrm{h}^{(t-1)} = h, z)}{\sum_{t=1}^{T} M(\mathrm{h}^{(t-1)} = h, z)}. \quad \text{(A.8)}$$

Now, defining n_y, $\eta_i^{(z)}$, $\gamma_{i,h}^{(z)}(t)$, and $\xi_{i,h,h'}^{(z)}(t)$ as in Algorithm 2.3, the update formulas for $\alpha_z^{(y)}$, $\pi_h^{(z)}$, and $A_{h,h'}^{(z)}$ become as defined in that algorithm.

Deriving the update equations for the means and covariances of the Gaussian emission distributions is not so simple, since these are not table CPDs and therefore the procedure we have just used is no longer valid. Nonetheless, the desired equations can be obtained by explicitly maximizing the ELBO with respect to these parameters. As we have seen in Section 1.4.3,

$$\text{ELBO} = \sum_{i=1}^{n} \mathbb{E}_{(\mathbf{h}_i, \mathrm{z}_i) \sim q} \log p(\boldsymbol{X}_i, \mathbf{h}_i, \mathrm{z}_i \mid y_i; \Theta) + \text{const} \quad \text{(A.9)}$$

where $q(\mathbf{h}_i, \mathrm{z}_i) = p(\mathbf{h}_i, \mathrm{z}_i \mid \boldsymbol{x}_i, y_i; \Theta^{(-)})$, being $\Theta^{(-)}$ the set of current values for all parameters. We are solely interested in maximizing the ELBO with respect to the parameters $\boldsymbol{\mu}_h^{(z)}$ and $\boldsymbol{\sigma}_h^{(z)}$, so we can plug in the expression for the joint $p(\boldsymbol{x}_i, \mathbf{h}_i, \mathrm{z}_i \mid y_i; \Theta)$ and ignore all terms that do not depend on these parameters:

$$\text{ELBO} = \sum_{i=1}^{n} \sum_{\mathbf{h}_i, \mathrm{z}_i} q(\mathbf{h}_i, \mathrm{z}_i) \log \left[p(\mathrm{z}_i \mid y_i) p(\mathrm{h}_i^{(0)} \mid \mathrm{z}_i) \prod_t p(\mathrm{h}_i^{(t)} \mid \mathrm{h}_i^{(t-1)}, \mathrm{z}_i) p(\boldsymbol{x}_i^{(t)} \mid \mathrm{h}_i^{(t)}, \mathrm{z}_i) \right]$$

$$= \sum_{i=1}^{n} \sum_{\mathrm{h}_i^{(t)}, \mathrm{z}_i} \sum_{t=1}^{T_i} q(\mathrm{h}_i^{(t)}, \mathrm{z}_i) \log p(\boldsymbol{x}_i^{(t)} \mid \mathrm{h}_i^{(t)}, \mathrm{z}_i) + \text{const.} \quad \text{(A.10)}$$

Now, all that remains is computing the gradient of the ELBO with respect to the parameters and solving for the critical points:

$$\begin{aligned}\nabla_{\boldsymbol{\mu}_h^{(z)}}\text{ELBO} &= \sum_{i=1}^{n}\sum_{t=1}^{T_i}\frac{q(\mathrm{h}_i^{(t)}=h,z)}{p(\boldsymbol{x}_i^{(t)}\mid \mathrm{h}_i^{(t)}=h,z)}\nabla_{\boldsymbol{\mu}_h^{(z)}}p(\boldsymbol{x}_i^{(t)}\mid \mathrm{h}_i^{(t)}=h,z)\\ &= \sum_{i=1}^{n}\sum_{t=1}^{T_i} q(\mathrm{h}_i^{(t)}=h,z)\frac{\boldsymbol{x}_i^{(t)}-\boldsymbol{\mu}_h^{(z)}}{{\boldsymbol{\sigma}_h^{(z)}}^2}, \qquad \text{(A.11)}\\ \nabla_{\boldsymbol{\sigma}_h^{(z)}}\text{ELBO} &= \sum_{i=1}^{n}\sum_{t=1}^{T_i}\frac{q(\mathrm{h}_i^{(t)}=h,z)}{p(\boldsymbol{x}_i^{(t)}\mid \mathrm{h}_i^{(t)}=h,z)}\nabla_{\boldsymbol{\sigma}_h^{(z)}}p(\boldsymbol{x}_i^{(t)}\mid \mathrm{h}_i^{(t)}=h,z)\\ &= \sum_{i=1}^{n}\sum_{t=1}^{T_i} q(\mathrm{h}_i^{(t)}=h,z)\left(\frac{(\boldsymbol{x}_i^{(t)}-\boldsymbol{\mu}_h^{(z)})^2}{{\boldsymbol{\sigma}_h^{(z)}}^3}-\frac{1}{\boldsymbol{\sigma}_h^{(z)}}\right), \qquad \text{(A.12)}\end{aligned}$$

where vector division and exponentiation should be interpreted as element-wise operations. Finally, solving for the critical points yields:

$$\begin{aligned}\boldsymbol{\mu}_h^{(z)} &= \frac{\sum_{i=1}^{n}\sum_{t=1}^{T_i} q(\mathrm{h}_i^{(t)}=h,z)\boldsymbol{x}_i^{(t)}}{\sum_{i=1}^{n}\sum_{t=1}^{T_i} q(\mathrm{h}_i^{(t)}=h,z)}\\ &= \frac{\sum_{i=1}^{n} p(z\mid \boldsymbol{X}_i,y_i;\Theta^{(-)})\sum_{t=1}^{T_i} p(\mathrm{h}^{(t)}=h\mid \boldsymbol{X}_i,z;\Theta^{(-)})\boldsymbol{x}_i^{(t)}}{\sum_{i=1}^{n} p(z\mid \boldsymbol{X}_i,y_i;\Theta^{(-)})\sum_{t=1}^{T_i} p(\mathrm{h}^{(t)}=h\mid \boldsymbol{X}_i,z;\Theta^{(-)})}, \qquad \text{(A.13)}\\ {\boldsymbol{\sigma}_h^{(z)}}^2 &= \frac{\sum_{i=1}^{n}\sum_{t=1}^{T_i} q(\mathrm{h}_i^{(t)}=h,z)(\boldsymbol{x}_i^{(t)}-\boldsymbol{\mu}_h^{(z)})^2}{\sum_{i=1}^{n}\sum_{t=1}^{T_i} q(\mathrm{h}_i^{(t)}=h,z)}\\ &= \frac{\sum_{i=1}^{n} p(z\mid \boldsymbol{X}_i,y_i;\Theta^{(-)})\sum_{t=1}^{T_i} p(\mathrm{h}^{(t)}=h\mid \boldsymbol{X}_i,z;\Theta^{(-)})(\boldsymbol{x}_i^{(t)}-\boldsymbol{\mu}_h^{(z)})^2}{\sum_{i=1}^{n} p(z\mid \boldsymbol{X}_i,y_i;\Theta^{(-)})\sum_{t=1}^{T_i} p(\mathrm{h}^{(t)}=h\mid \boldsymbol{X}_i,z;\Theta^{(-)})}. \qquad \text{(A.14)}\end{aligned}$$

Again, the formulas in Algorithm 2.3 follow by plugging $\eta_i^{(z)}$ and $\gamma_{i,h}^{(z)}(t)$ into the expressions above. □

A.1.2 EM for SpaMHMM (Algorithm 2.4)

We start by showing that the E-step is unaffected by the regularization term in equation (2.21) and then we will derive the update equations for the M-step.

As is a common practice in EM and variational methods, let $q(\mathrm{z},\mathbf{h})$ be an arbitrary distribution over the hidden variables of our model whose support contains the support

of p. Then,

$$J_r(\Theta) = \frac{1}{n}\sum_{i=1}^{n} \log \mathbb{E}_{(\mathbf{z}_i,\mathbf{h}_i)\sim q}\left[\frac{p(\boldsymbol{X}_i \mid y_i;\Theta)}{q(\mathbf{z}_i,\mathbf{h}_i)}\right] + \lambda R(\mathcal{G};\Theta)$$
$$\geq \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{(\mathbf{z}_i,\mathbf{h}_i)\sim q} \log\left[\frac{p(\boldsymbol{X}_i \mid y_i;\Theta)}{q(\mathbf{z}_i,\mathbf{h}_i)}\right] + \lambda R(\mathcal{G};\Theta). \tag{A.15}$$

where the inequality is a direct result of Jensen's inequality and $R(\mathcal{G};\Theta)$ is our regularization term, as defined in equation (2.21). We are therefore interested in maximizing this lower bound with respect to both Θ and q. Since $R(\mathcal{G};\Theta)$ does not depend on q, maximization with respect to this distribution while fixing $\Theta = \Theta^{(-)}$ yields $q(\mathbf{z}_i,\mathbf{h}_i) = p(\mathbf{z}_i,\mathbf{h}_i \mid \boldsymbol{X}_i, y_i;\Theta^{(-)})$, as for the case of standard (i.e. unregularized) EM. Thus,

$$J_r(\Theta) \geq \frac{1}{n}\text{ELBO}(\Theta) + \lambda R(\mathcal{G};\Theta) + \text{const}, \tag{A.16}$$

where the ELBO takes the exact same form as in equation (A.9).

Now all that remains is the maximization of this lower bound with respect to Θ. After noting that some parameters are constrained to sum up to 1, we could build the following Lagrangian and solve for the critical points:

$$L(\Theta,\Lambda) = \frac{1}{n}\text{ELBO}(\Theta) + \lambda R(\mathcal{G};\Theta) + \sum_{y=1}^{k}\mu_y\left(1 - ||\boldsymbol{\alpha}^{(y)}||_1\right)$$
$$+ \sum_{z=1}^{m}\sum_{h=1}^{s}\nu_h^{(z)}\left(1 - \sum_{h'=1}^{s}A_{h,h'}^{(z)}\right) + \sum_{z=1}^{m}\varrho_z\left(1 - ||\boldsymbol{\pi}_z||_1\right), \tag{A.17}$$

where $\Lambda \triangleq \bigcup_{h,y,z}\{\mu_y, \nu_h^{(z)}, \varrho_z\}$ is the set of all Lagrange multipliers. Since $R(\mathcal{G};\Theta)$ only depends on the mixture coefficients $\boldsymbol{\alpha}^{(y)}$, the update equations for all parameters except those are exactly the same as in Algorithm 2.3.

Unfortunately, setting $\nabla L = 0$ yields a system of equations that is non-linear in $\boldsymbol{\alpha}^{(y)}$ and no analytical solution can be found. However, we can resort to the reparametrization defined in equation (2.24) and update the new parameters $\boldsymbol{\beta}^{(y)}$ by performing gradient ascent over the unconstrained lower bound. Thus, computing the required gradient concludes the derivation of the algorithm.

Now we only care about the dependency of the ELBO on the mixture coefficients, so, following the same idea as in equation (A.10), we can write:

$$\begin{aligned}\text{ELBO} &= \frac{1}{n}\sum_{i=1}^{n}\sum_{z_i} q(z_i)\log p(z_i \mid y_i) + \text{const} \\ &= \frac{1}{n}\sum_{i=1}^{n}\sum_{z=1}^{m} q(z)\log \alpha_z^{(y_i)} + \text{const},\end{aligned} \tag{A.18}$$

and therefore,

$$\frac{\partial \text{ELBO}}{\partial \alpha_z^{(y)}} = \frac{1}{n}\sum_{i=1}^{n}\frac{q(z)}{\alpha_z^{(y)}}\mathbf{1}_{y_i=y}, \quad \frac{\partial R}{\partial \alpha_z^{(y)}} = \frac{1}{2}\sum_{\substack{y'=1,\\ y'\neq y}}^{k} G_{y,y'}\alpha_z^{(y')}. \tag{A.19}$$

Finally, by the chain rule,

$$\begin{aligned}\frac{\partial \text{ELBO}}{\partial \beta_z^{(y)}} &= \sum_{z'=1}^{m}\frac{\partial \text{ELBO}}{\partial \alpha_{z'}^{(y)}}\frac{\partial \alpha_{z'}^{(y)}}{\partial \beta_z^{(y)}} \\ &= \frac{1}{n}\sum_{i=1}^{n}\left(q(z) - \alpha_z^{(y)}\right)\mathbf{1}_{y_i=y\wedge\beta_z^{(y)}>0} \\ &= \frac{1}{n}\sum_{i=1}^{n}\left(p(z \mid \boldsymbol{X}_i, y_i; \Theta^{(-)}) - \alpha_z^{(y)}\right)\mathbf{1}_{y_i=y\wedge\beta_z^{(y)}>0},\end{aligned} \tag{A.20}$$

$$\begin{aligned}\frac{\partial R}{\partial \beta_z^{(y)}} &= \sum_{z'=1}^{m}\frac{\partial R}{\partial \alpha_{z'}^{(y)}}\frac{\partial \alpha_{z'}^{(y)}}{\partial \beta_z^{(y)}} \\ &= \alpha_z^{(y)}\sum_{\substack{y'=1,\\ y'\neq y}}^{k} G_{y,y'}\left(\alpha_z^{(y')} - \boldsymbol{\alpha}^{(y')^\top}\boldsymbol{\alpha}^{(y)}\right)\mathbf{1}_{\beta_z^{(y)}>0},\end{aligned} \tag{A.21}$$

and considering $\psi_z^{(y)}$, $\omega_z^{(y)}$, and $\delta_z^{(y)}$ as defined in Algorithm 2.4 and the gradient ascent update rule the formulas follow. □

A.2 Posterior distribution of observations

In this section, we show how to obtain the posterior distribution $p(\mathbf{X} \mid \boldsymbol{X}_{\text{pref}}, y)$ of sequences $\mathbf{X} \triangleq \left(\boldsymbol{x}^{(1)}, ..., \mathbf{x}^{(t)}\right)$ given an observed prefix sequence $\boldsymbol{X}_{\text{pref}} \triangleq \left(\boldsymbol{x}^{(-t_{\text{pref}}+1)}, ..., \boldsymbol{x}^{(0)}\right)$,

both coming from the graph node y. We start by writing this posterior as a marginalization with respect to the latent variable z:

$$\begin{aligned} p(\mathbf{X} \mid \boldsymbol{X}_{\text{pref}}, y) &= \sum_{\mathrm{z}} p(\mathbf{X}, \mathrm{z} \mid \boldsymbol{X}_{\text{pref}}, y) \\ &= \sum_{\mathrm{z}} p(\mathbf{X} \mid \boldsymbol{X}_{\text{pref}}, y, \mathrm{z}) p(\mathrm{z} \mid \boldsymbol{X}_{\text{pref}}, y) \\ &= \sum_{\mathrm{z}} p(\mathbf{X} \mid \boldsymbol{X}_{\text{pref}}, \mathrm{z}) p(\mathrm{z} \mid \boldsymbol{X}_{\text{pref}}, y), \end{aligned} \tag{A.22}$$

where the last equality follows from the fact that the observations $\mathbf{X}$ are conditionally independent of the graph node y given the latent variable z. The posterior $p(\mathrm{z} \mid \boldsymbol{X}_{\text{pref}}, y)$ may be obtained as done in Algorithm 2.3:

$$p(\mathrm{z} \mid \boldsymbol{X}_{\text{pref}}, y) \propto p(\boldsymbol{X}_{\text{pref}} \mid \mathrm{z}) p(\mathrm{z} \mid y). \tag{A.23}$$

We now focus on the computation of $p(\mathbf{X} \mid \boldsymbol{X}_{\text{pref}}, \mathrm{z})$. Let $\mathbf{h}_{\text{pref}} \triangleq \left(\mathrm{h}^{(-t_{\text{pref}}+1)}, ..., \mathrm{h}^{(-1)}\right)$ and $\mathbf{h} \triangleq \left(\mathrm{h}^{(0)}, ..., \mathrm{h}^{(t)}\right)$, then:

$$\begin{aligned} p(\mathbf{X} \mid \boldsymbol{X}_{\text{pref}}, \mathrm{z}) &= \sum_{\mathbf{h}_{\text{pref}}, \mathbf{h}} p(\mathbf{X}, \mathbf{h}_{\text{pref}}, \mathbf{h} \mid \boldsymbol{X}_{\text{pref}}, \mathrm{z}) \\ &= \sum_{\mathbf{h}_{\text{pref}}, \mathbf{h}} p(\mathbf{X} \mid \mathbf{h}_{\text{pref}}, \mathbf{h}, \boldsymbol{X}_{\text{pref}}, \mathrm{z}) p(\mathbf{h}_{\text{pref}}, \mathbf{h} \mid \boldsymbol{X}_{\text{pref}}, \mathrm{z}) \\ &= \sum_{\mathbf{h}_{\text{pref}}, \mathbf{h}} p(\mathbf{X} \mid \mathbf{h}, \mathrm{z}) p(\mathbf{h}_{\text{pref}} \mid \mathbf{h}, \boldsymbol{X}_{\text{pref}}, \mathrm{z}) p(\mathbf{h} \mid \boldsymbol{X}_{\text{pref}}, \mathrm{z}) \\ &= \sum_{\mathbf{h}} p(\mathbf{X} \mid \mathbf{h}, \mathrm{z}) p(\mathbf{h} \mid \boldsymbol{X}_{\text{pref}}, \mathrm{z}) \\ &= \sum_{\mathbf{h}} p(\mathrm{h}^{(0)} \mid \boldsymbol{X}_{\text{pref}}, \mathrm{z}) \prod_{\tau=1}^{t} p(\mathrm{h}^{(\tau)} \mid \mathrm{h}^{(\tau-1)}, \mathrm{z}) p(\mathbf{x}^{(\tau)} \mid \mathrm{h}^{(\tau)}, \mathrm{z}), \end{aligned} \tag{A.24}$$

where we have used the independence assumptions that characterize the HMM. Here, the initial state posteriors $p(\mathrm{h}^{(0)} \mid \boldsymbol{X}_{\text{pref}}, \mathrm{z})$ are actually the final state posteriors for the sequence $\boldsymbol{X}_{\text{pref}}$ for each HMM in the mixture, so they can be computed as in Algorithm 1.1.

Thus, we conclude that inference in the posterior model $p(\mathbf{X} \mid \boldsymbol{X}_{\text{pref}}, y)$ corresponds to inference in a (Spa)MHMM where the original mixture coefficients and initial state probabilites are replaced with their respective posteriors, $p(\mathrm{z} \mid \boldsymbol{X}_{\text{pref}}, y)$ and $p(\mathrm{h}^{(0)} \mid \boldsymbol{X}_{\text{pref}}, \mathrm{z})$. All remaining parameters are unchanged.

Appendix B

Tackling unsupervised domain adaptation with optimism and consistency – supplementary material

B.1 Proof of Theorem 3.4

The presented bound is an immediate consequence of two theorems of [73, 75], which we present here as lemmas:

Lemma B.1. *(Lemma 4 from Blitzer et al. [75]) Let $\mathcal{H}$ be a hypothesis class with VC-dimension d. For each $j \in \{1, 2, ..., k\}$, consider a labeled set of n/k samples drawn from the source domain $\mathcal{S}_j$. For any $h \in \mathcal{H}$ and any $\boldsymbol{\alpha} \in \Delta$, with probability at least $1 - \delta$ over the choice of samples,*

$$|\widehat{\epsilon}_{\boldsymbol{\alpha}}(h) - \epsilon_{\boldsymbol{\alpha}}(h)| \leq 2\sqrt{\frac{k}{n}\left(2d\log(2(n+1)) + \log\frac{4}{\delta}\right)\sum_{j=1}^{k}\alpha_j^2}. \tag{B.1}$$

Lemma B.2. *(Theorem 2 from Ben-David et al. [73]) Let $\mathcal{H}$ be a hypothesis class with VC-dimension d. Consider n unlabeled samples drawn from each of the two domains $\mathcal{S}$ (source) and $\mathcal{T}$ (target). Then, for every $h \in \mathcal{H}$, with probability at least $1 - \delta$ over the choice of samples,*

$$\epsilon_{\mathcal{T}}(h) \leq \epsilon_{\mathcal{S}}(h) + \frac{1}{2}\widehat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}, \mathcal{T}) + 2\sqrt{\frac{1}{n}\left(2d\log(2n) + \log\frac{2}{\delta}\right)} + \lambda, \tag{B.2}$$

where $\lambda = \min_{h\in\mathcal{H}} \epsilon_{\mathcal{S}}(h) + \epsilon_{\mathcal{T}}(h)$.

Proving our result is now straightforward. We can apply Lemma B.2 taking $\mathcal{S}_{\boldsymbol{\alpha}}$ as the source domain. Note that $\epsilon_{S_{\boldsymbol{\alpha}}}(h) = \sum_{j=1}^{k} \alpha_j \epsilon_{S_j}(h)$ for any $h \in \mathcal{H}$, and therefore $\lambda_{\boldsymbol{\alpha}} =$

$\min_{h \in \mathcal{H}} \sum_{j=1}^{k} \alpha_j \epsilon_{\mathcal{S}_j}(h) + \epsilon_{\mathcal{T}}(h)$. Combining the obtained bound with Lemma B.1 through the union bound yields the desired result. □

B.2 Model overview

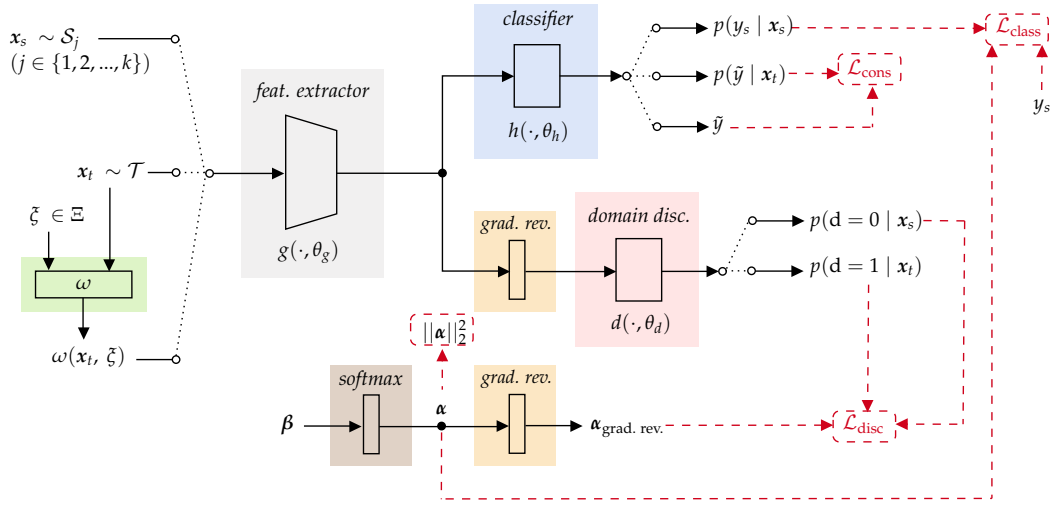


FIGURE B.1: Block diagram representing the proposed model (MODA-FM).

To enhance clarity, a detailed schematic of our model and loss terms is shown in Figure B.1. The three main blocks are the feature extractor $g(\cdot, \boldsymbol{\theta}_g)$, the classifier $h(\cdot, \boldsymbol{\theta}_h)$, and the domain discriminator $d(\cdot, \boldsymbol{\theta}_d)$. It is worth noting the usage of gradient reversal layers at the input of the domain discriminator and before the $\boldsymbol{\alpha}$ that is used in the discriminator loss $\mathcal{L}_{\text{disc}}$. As explained in Section 3.3.2.1, the usage of this layer allows the model to be trained using standard backpropagation and stochastic gradient descent (or any of its variants) over all model parameters.

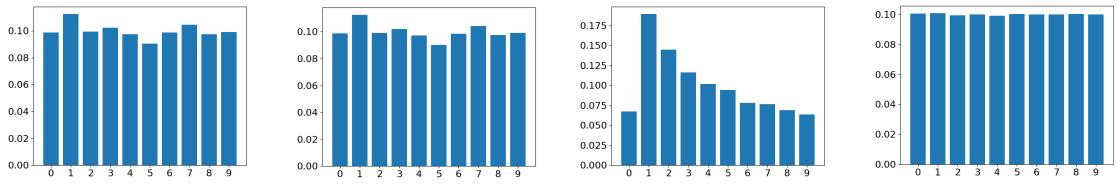
B.3 Experiments – further results and details

B.3.1 Label distributions

As remarked throughout this work and formally discussed and proven in [79], having different marginal label distributions across source and target domains poses difficulties to the DA task and, under this scenario, domain-invariant representations can increase the error on the target domain. Since we claim that the adopted consistency regularization can help mitigate this issue, it is relevant to observe the distributions of labels across

domains in the datasets used in this work, which are shown in Figures B.2, B.3, and B.4.* We also provide the pairwise Jensen-Shannon distances for the label distributions of all domains in Tables B.1, B.2, B.3, and B.4. The Jensen-Shannon distance is convenient since it is a metric (so it is symmetric, non-negative and zero if and only if the two distributions coincide) and upper bounded by $\sqrt{\log 2}\,(\approx 0.8326)$. Therefore, it provides a comprehensive measure to evaluate the dissimilarity between two distributions.

In the digits datasets (Figure B.2, Table B.1), MNIST and MNIST-M have very similar label distributions and SynthDigits has an almost uniform label´ distribution. SVHN has the most skewed distribution, which is radically different from the remaining. Thus, this is the most challenging domain from this perspective. In Office-31 (Figure B.3, Table B.2), label distributions across domains differ significantly. In the Amazon Reviews dataset (Figure B.4, Table B.3), the label distribution is almost uniform for all domains. Finally, in DomainNet (Table B.4), the label distributions are highly dissimilar across all domains. Interestingly, domains *inf* and *pnt* are the most distant from all remaining domains and *qdr* is the closest one.



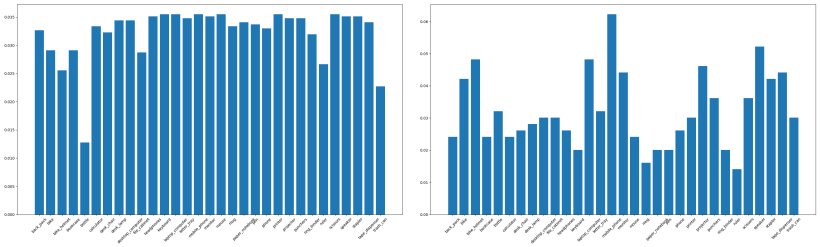
(A) MNIST (B) MNIST-M (C) SVHN (D) SynthDigits

FIGURE B.2: Label distributions in the digits datasets.

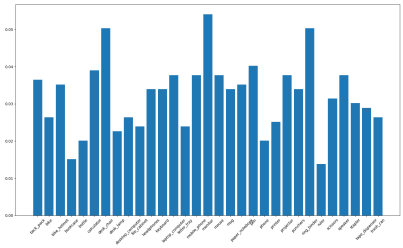
	MNIST	MNIST-M	SVHN	SynthDigits
MNIST	0	$\underline{2.73 \cdot 10^{-4}}$	$\underline{1.17 \cdot 10^{-1}}$	$\underline{1.83 \cdot 10^{-2}}$
MNIST-M	$\underline{2.73 \cdot 10^{-4}}$	0	$\underline{1.17 \cdot 10^{-1}}$	$1.84 \cdot 10^{-2}$
SVHN	$\mathbf{1.17 \cdot 10^{-1}}$	$\mathbf{1.17 \cdot 10^{-1}}$	0	$\mathbf{1.26 \cdot 10^{-1}}$
SynthDigits	$1.83 \cdot 10^{-2}$	$1.84 \cdot 10^{-2}$	$\mathbf{1.26 \cdot 10^{-1}}$	0
Average	$4.51 \cdot 10^{-2}$	$4.52 \cdot 10^{-2}$	$1.20 \cdot 10^{-1}$	$5.42 \cdot 10^{-2}$

TABLE B.1: Jensen-Shannon distances between the label distributions of each pair of digits domains. On each column, the largest distance is in bold and the smallest is underlined.

*The histogram for the DomainNet dataset is not provided since the number of classes (345) is too large, making the plot barely interpretable.



(A) Amazon (B) DSLR

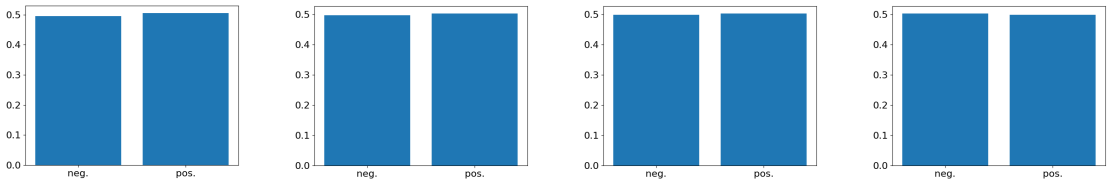


(C) Webcam

FIGURE B.3: Label distributions in Office-31.

	Amazon	DSLR	Webcam
Amazon	0	$1.33 \cdot 10^{-1}$	$9.76 \cdot 10^{-2}$
DSLR	$\mathbf{1.33 \cdot 10^{-1}}$	0	$\mathbf{1.45 \cdot 10^{-1}}$
Webcam	$9.76 \cdot 10^{-2}$	$\mathbf{1.45 \cdot 10^{-1}}$	0
Average	$1.15 \cdot 10^{-1}$	$1.39 \cdot 10^{-1}$	$1.21 \cdot 10^{-1}$

TABLE B.2: Jensen-Shannon distances between the label distributions of each pair of domains in Office-31. On each column, the largest distance is in bold.



(A) Books (B) DVD (C) Electronics (D) Kitchen

FIGURE B.4: Label distributions in Amazon Reviews.

	Books	DVD	Electronics	Kitchen
Books	0	$1.67 \cdot 10^{-3}$	$1.93 \cdot 10^{-3}$	$\mathbf{5.09 \cdot 10^{-3}}$
DVD	$\underline{1.67 \cdot 10^{-3}}$	0	$\underline{2.53 \cdot 10^{-4}}$	$3.42 \cdot 10^{-3}$
Electronics	$1.93 \cdot 10^{-3}$	$\underline{2.53 \cdot 10^{-4}}$	0	$\underline{3.17 \cdot 10^{-3}}$
Kitchen	$\mathbf{5.09 \cdot 10^{-3}}$	$\mathbf{3.42 \cdot 10^{-3}}$	$\mathbf{3.17 \cdot 10^{-3}}$	0
Average	$2.90 \cdot 10^{-3}$	$1.78 \cdot 10^{-3}$	$1.78 \cdot 10^{-3}$	$3.89 \cdot 10^{-3}$

TABLE B.3: Jensen-Shannon distances between the label distributions of each pair of domains in Amazon Reviews. On each column, the largest distance is in bold and the smallest is underlined.

	clp	*inf*	*pnt*	*qdr*	*rel*	*skt*
clp	0	$3.14 \cdot 10^{-1}$	$3.34 \cdot 10^{-1}$	$1.98 \cdot 10^{-1}$	$2.31 \cdot 10^{-1}$	$2.60 \cdot 10^{-1}$
inf	$3.14 \cdot 10^{-1}$	0	$\mathbf{3.79 \cdot 10^{-1}}$	$\mathbf{2.83 \cdot 10^{-1}}$	$\mathbf{3.03 \cdot 10^{-1}}$	$\mathbf{3.26 \cdot 10^{-1}}$
pnt	$\mathbf{3.34 \cdot 10^{-1}}$	$\mathbf{3.79 \cdot 10^{-1}}$	0	$2.77 \cdot 10^{-1}$	$2.99 \cdot 10^{-1}$	$3.10 \cdot 10^{-1}$
qdr	$\underline{1.98 \cdot 10^{-1}}$	$\underline{2.83 \cdot 10^{-1}}$	$\underline{2.77 \cdot 10^{-1}}$	0	$\underline{1.35 \cdot 10^{-1}}$	$\underline{2.27 \cdot 10^{-1}}$
rel	$2.31 \cdot 10^{-1}$	$3.03 \cdot 10^{-1}$	$2.99 \cdot 10^{-1}$	$\underline{1.35 \cdot 10^{-1}}$	0	$2.67 \cdot 10^{-1}$
skt	$2.60 \cdot 10^{-1}$	$3.26 \cdot 10^{-1}$	$3.10 \cdot 10^{-1}$	$2.27 \cdot 10^{-1}$	$2.67 \cdot 10^{-1}$	0
Average	$2.67 \cdot 10^{-1}$	$3.21 \cdot 10^{-1}$	$3.20 \cdot 10^{-1}$	$2.24 \cdot 10^{-1}$	$2.47 \cdot 10^{-1}$	$2.78 \cdot 10^{-1}$

TABLE B.4: Jensen-Shannon distances between the label distributions of each pair of domains in the DomainNet dataset. On each column, the largest distance is in bold and the smallest is underlined.

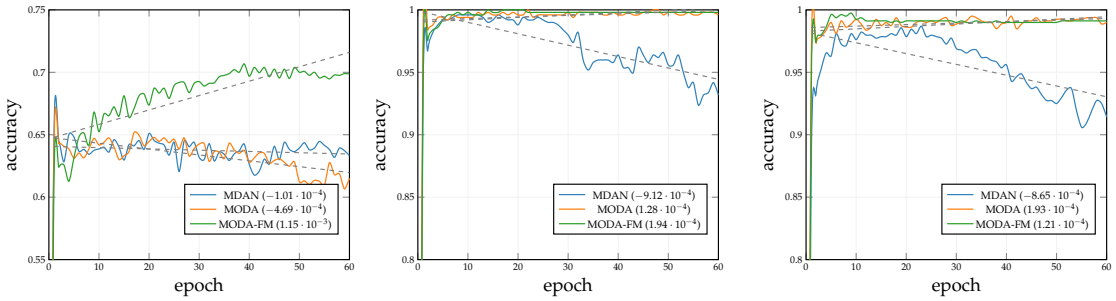
B.3.2 Effect of over-training

When the label distributions differ across domains, training the feature extractor and domain discriminator for a large number of iterations tends to lead to an increased target error. This is a direct effect of the curse of domain-invariant representations and it has been verified experimentally in [79].

In this experiment, we want to evaluate if the increased robustness of the feature extractor provided by the adopted consistency regularization helps to mitigate this issue. For this purpose, we use Office-31 since it is fairly small and therefore easy to overtrain and the label distributions across domains are significantly skewed (see B.3.1). We train our model and two baselines (MDAN [72] and MODA) for 60 epochs and we observe the evolution of the model accuracy on the target data. The plots are shown in Figure B.5. As we see there, in MODA-FM the accuracy keeps stable after reaching the maximum in all domains. Contrarily, in MODA and especially in MDAN, the accuracy tends to decay after reaching the maximum. These observations strongly suggest that MODA-FM succeeds on mitigating the curse of domain-invariant representations. In MODA the problem is less pronounced than in MDAN probably because the latter will continue optimizing itself until it can produce domain-invariant representations for all source domains, whereas the former will simply ignore the hardest source domains by assigning them a low weight (possibly even zero).

B.3.3 Hyperparameter sensitivity analysis

Our model comprises three main hyperparameters: μ_d, μ_s and μ_c. The hyperparameter μ_d controls the relative weight of the domain discriminator loss and is present in every work on adversarial DA, thus its effect has already been studied extensively. For this reason, we focus on the effect of μ_s and μ_c. We use the digits datasets and evaluate the accuracy



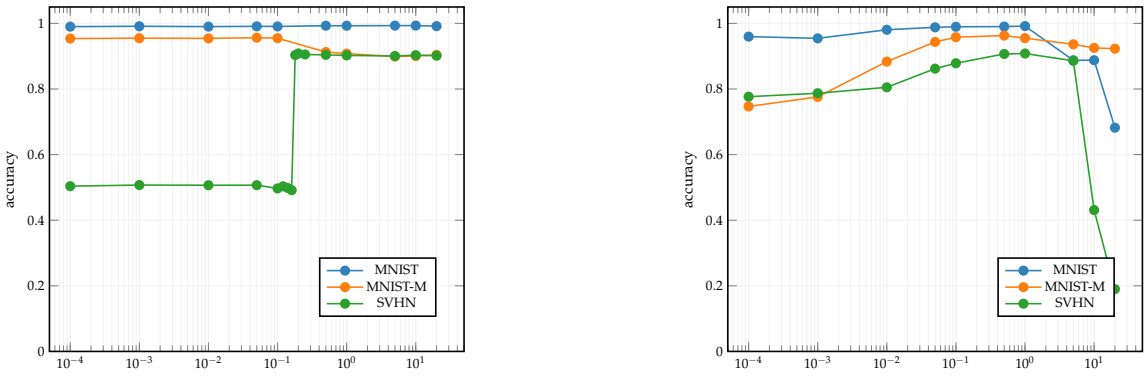
(A) Amazon (B) DSLR (C) Webcam

FIGURE B.5: Test accuracy over 60 training epochs for our model and two baselines in Office-31. The tendency line for each curve is also shown (dashed lines) and the respective slope is indicated in brackets in each plot legend. The domain indicated below each plot is the target.

on each target domain while varying either μ_s or μ_c and keeping the other one constant at the optimal value.

A sufficiently large value of μ_s forces $\boldsymbol{\alpha}$ to converge to a vector with all components equal to $1/k$, weighting all source domains equally. Setting μ_s close to zero corresponds to the most optimistic scenario: the sparsity penalization is dropped and so, after sufficiently many training iterations, $\boldsymbol{\alpha}$ would converge to a one-hot vector, choosing the source domain with the minimum difference of classification and discrimination losses ($\mathcal{L}_{class} - \mu_d \mathcal{L}_{disc}$). Figure B.6a shows that, for MNIST, similar results are obtained with any of those strategies. The fact that digits classification in MNIST is significantly easier than in any of the remaining datasets likely explains this behavior. Domain adaptation for MNIST-M is slightly favored by smaller values of μ_s, meaning that a more optimistic approach works better here. For SVHN, the opposite holds and the results are very poor when low sparsity regularization is employed. There is even a strong discontinuity in the accuracy plot for this domain around $\mu_s \approx 0.2$. This discontinuity divides situations where $\boldsymbol{\alpha}$ collapses into the easiest source domain ($\mu_s < 0.2$) and those where it does not collapse ($\mu_s > 0.2$). We elaborate more on this in B.3.4. In all cases, we observe that a $\mu_s \in [0.2, 0.6]$ provides near-optimal results.

The effect of varying μ_c is plotted in Figure B.6b. Values of μ_c close to zero drop the consistency regularization and, therefore, we obtain results close to those of MODA. Significant performance gains are obtained for $\mu_c > 0.01$ and the optimal is reached for $\mu_c \in [0.4, 1.0]$ for all domains.



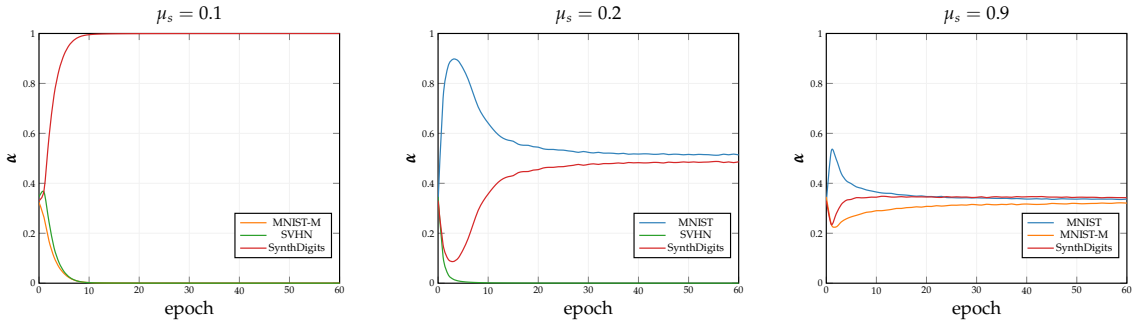
(A) μ_s (B) μ_c

FIGURE B.6: Test accuracy as a function of hyperparameters μ_s (a) and μ_c (b) using the digits datasets. The domain corresponding to each line is the target.

B.3.4 Evolution of the source weights

Here, we study the evolution of the source weights $\boldsymbol{\alpha}$ over training. The behavior is determined by the evolution of the classification and discrimination losses of each source domain and also by the choice of the hyperparameter μ_s. The effect of varying μ_s on the model performance was analyzed in B.3.3. Now, we are interested in observing the dynamics of $\boldsymbol{\alpha}$ over the training epochs, for different source domains, using the corresponding optimal μ_s. We use the digits datasets for this purpose and we present the results in Figure B.7.

For MNIST, where the lowest μ_s is used, we observe that the mixture coefficients rapidly collapse into the easiest source domain (SynthDigits). Nonetheless, in B.3.3 we have observed that the target accuracy for MNIST was almost insensitive to the choice of μ_s, meaning that using only the data from the easiest source domain or weighting all source domains equally would produce identical results. When we take MNIST-M and SVHN as target domains, with a slightly increased μ_s, a different behavior occurs. Early in the training process, the weight for the easiest source domain (MNIST) increases rapidly, following the fast decrease in the corresponding loss. Later, as the loss for the easiest source domain plateaus and the remaining keep decreasing, the weights for the remaining active source domains start to increase. This behavior explains the discontinuity we have observed in the plot of target accuracy vs. μ_s for SVHN (Figure B.6a): if μ_s is sufficiently small to allow $\boldsymbol{\alpha}$ to rapidly collapse into MNIST, the data of the remaining source domains is simply discarded for the remaining of the training process and the corresponding weights will never increase.



(A) MNIST (B) MNIST-M (C) SVHN

FIGURE B.7: Source weights $\boldsymbol{\alpha}$ for each domain over 60 training epochs in the digits datasets. The target domain and the value of μ_s that was used are indicated below and above each plot, respectively.

B.3.5 Network architectures

The following notation is used to designate network layers: `Conv(n, k)` – 2-D convolutional layer with `n` output channels, square kernels with size `k`, and unit stride, followed by a rectified linear activation; `MaxPool(k)` – 2-D max-pooling over non-overlapping regions of `k` × `k` pixels; `AdaptAvgPool(k)` – 2-D adaptive average pooling where the output size is `k` × `k` pixels. `FC(n)` – fully-connected layer with `n` output neurons, followed by a rectified linear activation except for output layers; `Dropout(p)` – dropout layer where `p` is the probability of zeroing an element. A gradient reversal layer is present at the input of the domain discriminator in all models. All classifiers and domain discriminators are followed by a softmax layer that maps the output to the corresponding class probabilities.

Digits classification Input images have shape 3 × 32 × 32. Feature extractor: `Conv(64, 3)` → `MaxPool(2)` → `Conv(128, 3)` → `MaxPool(2)` → `Conv(256, 3)`. Task classifier: `MaxPool(2)` → `Conv(256, 3)` → `FC(2048)` → `FC(1024)` → `FC(10)`. Domain discriminator: `MaxPool(2)` → `FC(2048)` → `FC(2048)` → `FC(1024)` → `FC(2)`.

Object classification (Office-31 dataset) Input images have shape 3 × 256 × 256. Feature extractor: ResNet-50 up to stage 3. Task classifier / domain discriminator: ResNet-50 stage 4 → `AdaptAvgPool(1)` → `FC(#classes)`, where `#classes` = 31 for the task classifier and `#classes` = 2 for the domain discriminator.

Sentiment analysis (Amazon Reviews dataset) Input features have dimension 5000. Feature extractor: `Dropout(p)` → `FC(1000)` → `Dropout(p)` → `FC(500)` → `Dropout(p)` →

`FC(100)` $\rightarrow$ `Dropout(p)`, where `p` $= 0$ except for producing the augmented samples used in the consistency regularization. Task classifier / domain discriminator: `FC(2)`.

Large scale object classification (DomainNet dataset) Input images have shape $3 \times 192 \times 192$. Feature extractor: ResNet-152 up to stage 3. Task classifier / domain discriminator: ResNet-152 stage 4 $\rightarrow$ `AdaptAvgPool(1)` $\rightarrow$ `FC(#classes)`, where `#classes` $= 345$ for the task classifier and `#classes` $= 2$ for the domain discriminator.

B.3.6 Choice of hyperparameters

Table B.5 presents the values assigned to each hyperparameter. For some hyperparameters in a given dataset, a set or range of values is presented instead of a single value, meaning that the value for that hyperparameter was selected via cross-validation on the given set/range. This cross-validation consisted on 20 iterations of random search for each experiment, where no data from the target domain was used and each source domain was taken as target in turn. The hyperparameter combination that yielded the best average result was chosen.

Hyperparameter	Digits	Office-31	Amazon Reviews	DomainNet
μ_d (8)	$[10^{-4}, 10^{0}]$	$[10^{-4}, 10^{0}]$	$[10^{-4}, 10^{0}]$	10^{-2}
μ_s (8)	$[10^{-5}, 10^{0}]$	10^{-2}	$[10^{-6}, 10^{-1}]$	10^{-1}
μ_c (16)	$[10^{-2}, 10^{0}]$	$[10^{-2}, 10^{0}]$	$[10^{-2}, 10^{0}]$	10^{-1}
τ (15)	0.9	0.9	0.9	0.9
optimizer	AdaDelta	AdaDelta	AdaDelta	AdaDelta
no. epochs	$\{10, 20, \ldots, 60\}$	$\{10, 15, \ldots, 60\}$	$\{5, 10, \ldots, 40\}$	100
learning rate	0.1	0.1	1.0	0.01
batch size	8	8	20	8
RandAugment: n	2	2	n.a.	2
RandAugment: m_{min}	3	3	n.a.	3
RandAugment: m_{max}	10	10	n.a.	10
dropout: p_{min}	n.a.	n.a.	0.2	n.a.
dropout: p_{max}	n.a.	n.a.	0.8	n.a.

TABLE B.5: Values and search ranges for all hyperparameters in all experiments.

B.3.7 Image transformations

The list of image transformations used in RandAugment is provided in Table B.6. The cutout transformation is the first transformation to be applied to every image. This transformation occludes a randomly located square region with a length equal to 30% of the image length. The remaining n transformations are chosen at random from the provided

list. All transformations were normalized such that a transformation of magnitude $m = 0$ corresponds to the identity (i.e. unchanged image) and a magnitude $m = 30$ (maximum admissible value) corresponds to the maximum distortion. We implemented the image transformations using the Python Imaging Library (PIL).

Transformation	Range	Transformation	Range
AutoContrast	$\{0, 1\}$	Rotate	$[-30°, 30°]$
Brightness	$[0.1, 1.9]$	Sharpness	$[0.1, 1.9]$
Color	$[0.1, 1.9]$	ShearX	$[0, 0.3]$
Contrast	$[0.1, 1.9]$	ShearY	$[0, 0.3]$
Equalize	$\{0, 1\}$	Solarize	$[0, 255]$
FlipX*	$\{0, 1\}$	TranslateX	$[-0.45, 0.45]$
Invert	$\{0, 1\}$	TranslateY	$[-0.45, 0.45]$
Posterize	$[4, 8]$		

TABLE B.6: RandAugment image transformations and respective ranges.

B.3.8 Sample images

We present sample images from all image datasets used in our experiments in Figures B.8, B.9, and B.10. The result of the RandAugment transformation is also shown for various values of the parameter m.

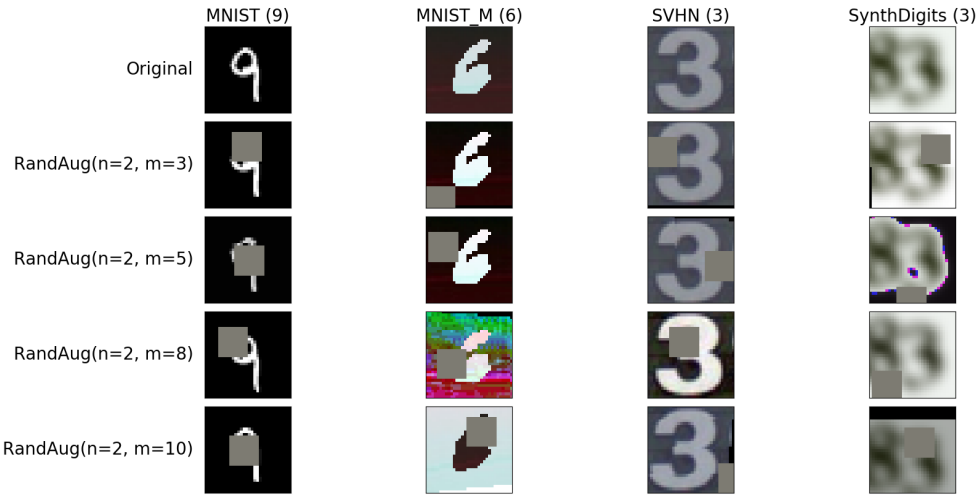


FIGURE B.8: Sample original and transformed images from the digits datasets. The ground-truth label is in brackets.

*Excluded in the digits experiment since it is not label-preserving.

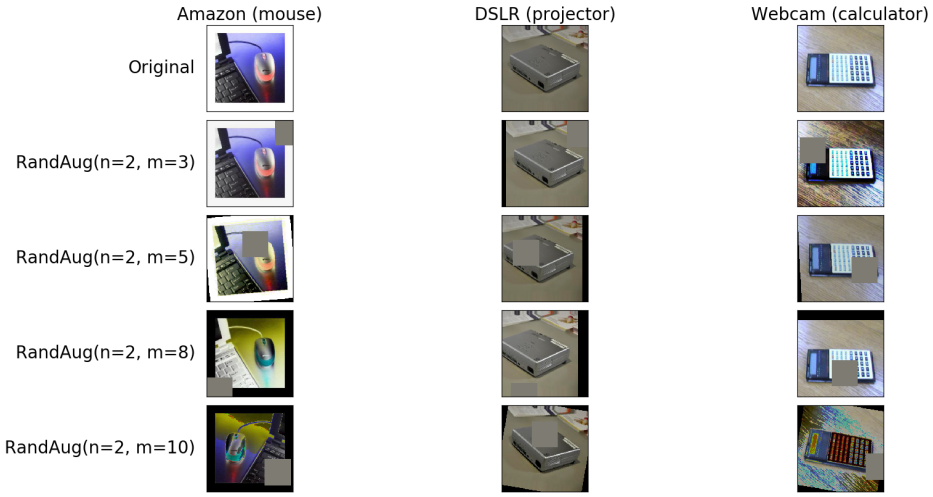


FIGURE B.9: Sample original and transformed images from each domain in the Office-31 dataset. The ground-truth label is in brackets.

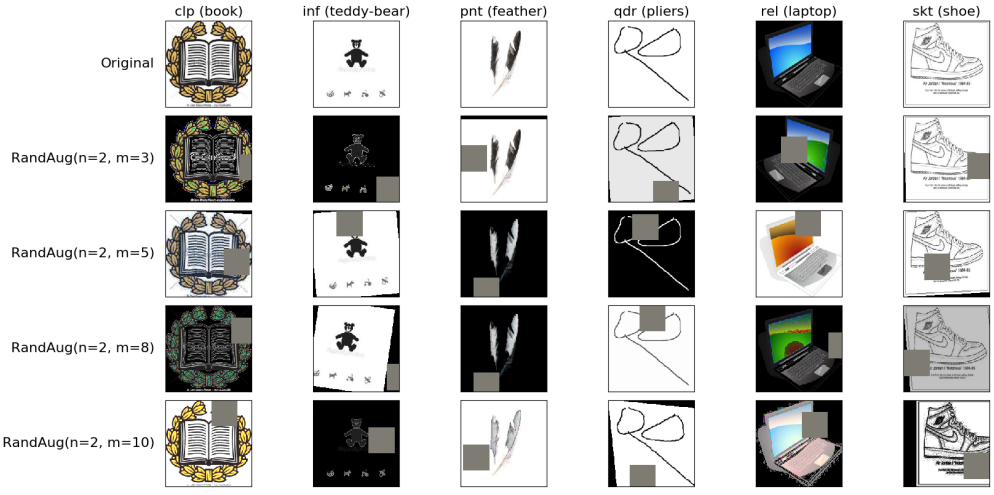


FIGURE B.10: Sample original and transformed images from each domain in the DomainNet dataset. The ground-truth label is in brackets.

Appendix C

DeSIRe: Deep Signer-Invariant Representations for Sign Language Recognition – supplementary material

C.1 Architecture

As shown in Figure C.1, the architecture of DeSIRe comprises a CVAE and a classifier.

C.1.1 CVAE

The CVAE consists of an encoder and a decoder. The encoder network attempts to learn a stochastic mapping from an input image $\boldsymbol{X}$, its class label y, and signer identity s to a latent representation $\boldsymbol{z}$. We condition the encoder on y and s by simply concatenating the class and signer identity as extra channels in the input image. In this case, both y and s are represented categorically using one-hot encoding. The encoder network then consists of a sequence of several 3×3 convolutional layers with batch-normalization and leaky rectified linear units (LeakyReLUs) as non-linearities. For downsampling, the stride length of every convolution is set to 2. On top of that, there are two output fully connected (a.k.a. dense) layers, with linear activation functions, describing the mean $\boldsymbol{\mu}_e(\mathbf{X}, \mathrm{y}, \mathrm{s}; \boldsymbol{\theta}_e)$ and the log-variance $\log \sigma_e^2(\mathbf{X}, \mathrm{y}, \mathrm{s}; \boldsymbol{\theta}_e)$ of the latent space distribution $q(\mathbf{z} \mid \mathbf{X}, \mathrm{y}, \mathrm{s}; \boldsymbol{\theta}_e)$.

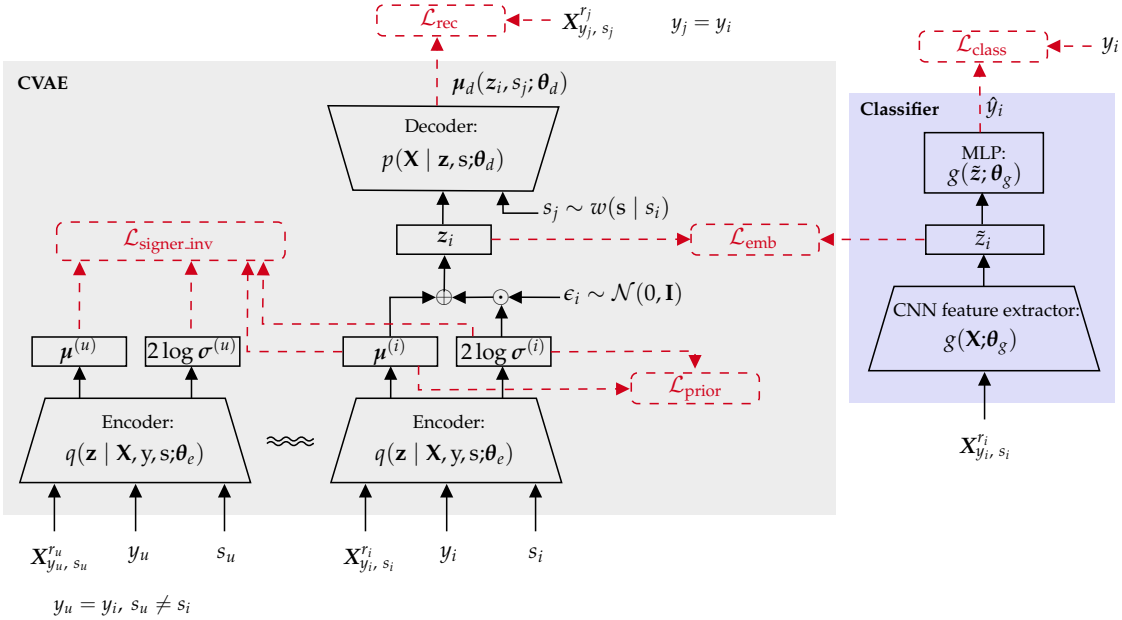


FIGURE C.1: The architecture of the proposed DeSIRe deep neural network for signer-independent SLR. It comprises two main modules or components: a conditional variational atoencoder (CVAE) and a classifier.

The decoder module will then generate a latent code $\boldsymbol{z}$ sampled from $q(\mathbf{z} \mid \boldsymbol{X}, y, s; \boldsymbol{\theta}_e)$ and proceed to the reconstruction of the original input $\boldsymbol{X}$. In practice, the latent code $\boldsymbol{z}$ is concatenated with a one-hot representation of the signer identity s to be fed to the decoder network. The decoder network comprises several 2-D transposed convolutions for upsampling and densifying the incoming activations. Every transposed convolutional layer is followed by batch-normalization and a LeakyReLU. The output layer also consists of a transposed convolutional layer but with a hyperbolic tangent activation function in order to output the reconstruction $\boldsymbol{\mu}_d(z, s; \boldsymbol{\theta}_d)$ of the normalized input $\boldsymbol{X}$.

C.1.2 Classifier

The implemented classifier module follows a typical CNN architecture for classification tasks. It starts with a block of convolutional layers for feature extraction purposes, producing representations $\tilde{\boldsymbol{z}} \triangleq g(\boldsymbol{X}; \boldsymbol{\theta}_g)$. This is followed by a block of fully-connected layers for sign classification, which predicts the sign class $\hat{y} \triangleq h(\tilde{\boldsymbol{z}}; \boldsymbol{\theta}_h)$. In particular, the convolutional block comprises a sequence of several pairs of consecutive 3×3 convolutional layers with ReLUs as non-linearities. For downsampling, the last convolutional layer of each pair has a stride of 2.

The fully-connected block consists of a sequence of fully-connected layers with ReLUs as the non-linear functions. The last fully-connected layer has a softmax activation function which outputs the probabilities for each sign class.

C.2 Training strategies

We have observed experimentally that the classification task is much easier than the reconstruction task of the CVAE. That is, the classifier tends to overfit the data with fewer training epochs than the CVAE, learning embeddings that are essentially not signer-invariant. In order to avoid this behavior, we have adopted an annealing strategy to define the classification weight λ_2. Specifically, at the start of training, this weight is set to zero, so that the CVAE learns to produce signer-invariant latent representations. At this stage, the CVAE behaves as a pure teacher model for the classifier network and, therefore, the L_{emb} error is backpropagated only through the classifier. After a few epochs, the weight λ_2 starts increasing according to a sigmoid annealing schedule and the L_{emb} loss starts to be backpropagated through the CVAE encoder as well. This procedure will endow the CVAE with a better sense of the classification task and hence promote class separability in the latent space. As a result, the model will be able to learn signer-invariant representations that are in fact highly discriminative for the sign recognition task.

Following Bowman et al. [201] and in order to stabilize the training of the CVAE, we have employed a similar annealing strategy to define the KL divergence weights of the prior and signer-invariant loss terms, α_1 and α_2, respectively.

C.3 Implementation details

All deep models were implemented in PyTorch and trained with the Adam optimization algorithm using a batch size of 32 samples. For reproducibility purposes, the source code as well as the weights of the trained models are publicly available online*.

The hyperparameters that are common to all the implemented models (i.e., the learning rate and the L^2 regularization coefficient) as well as some hyperparameters that are specific to the DeSIRe model (i.e. λ_1, λ_2, α_1 and α_2) and to the implemented baseline 2 (i.e. γ) were optimized by means of a grid search approach and cross-validation on the training set. The hyperparameter ρ in $w(\mathrm{s} \mid s_i)$, defined for the proposed sampling scheme

*URL will be made available after the decision.

Hyperparameters	Symbol	Set
Leaning rate	–	$\{1e^{-03}, 1e^{-04}\}$
L^2-norm coefficient	–	$\{1e^{-04}, 1e^{-05}\}$
L_{triplet} weight	γ	{0.1,0.5,1,5,10}
L_{emb} weight	λ_1	{0.1,0.5}
L_{class} weight	λ_2	{1,5,10}
L_{prior} weight	α_1	$\{5e^{-03}, 8e^{-02}\}$
$L_{\text{signer\_inv}}$ weight	α_2	$\{8e^{-02}, 4e^{-01}, 8e^{-01}\}$

TABLE C.1: Hyperparameter sets for the DeSIRe model and baselines.

of the signer identity, was set to $\rho = 0.5$. The dropout rate was empirically set to 0.5 for all the experiments. The set of values of the adopted hyperparameters grid search is presented in Table C.1.

During the training stage of all the implemented models, besides L^2 regularization and dropout, a randomized data augmentation scheme was also employed. Following Ferreira et al. [180], the adopted data augmentation procedure is based on both geometric and color transformations. The underlying idea is to further increase the robustness of the models to the wide range of hand gestures positions, poses, viewing angles as well as to different illumination conditions and contrasts.

A detailed description of the architecture of the proposed model is presented in Table C.2. For illustrative purposes, the presented DeSIRe architecture considers input colour images with a resolution of 100×100 pixels, 10 signer identities in the training set and a total of 10 sign classes. It is important to stress out that, for a fair comparison, the topology of both implemented baselines follows the same architecture of the classifier component of the proposed DeSIRe model.

Layer #	DeSIRe module	Layer type	Non-linearity	Output shape	Connected to
-	Inputs	input_x	-	(3,100,100)	-
-		input_y_2d	-	(10,100,100)	-
-		input_s_2d	-	(10,100,100)	-
-		input_s_1d	-	(10,)	-
1	$q(\mathbf{z} \mid \mathbf{X}, y, s; \boldsymbol{\theta}_e)$	Concat2d-1	-	(23,100,100)	[input_x; input_y_2d; input_s_2d]
2		Conv2d-1	LeakyReLU	(64,50,50)	Concat2d-1
3		Conv2d-2	LeakyReLU	(64,25,25)	Conv2d-1
4		Conv2d-3	LeakyReLU	(128,13,13)	Conv2d-2
5		Conv2d-4	LeakyReLU	(256,7,7)	Conv2d-3
6		Conv2d-5	LeakyReLU	(512,4,4)	Conv2d-4
7		Dense-1	Linear	(128,)	Conv2d-5
8		Dense-2	Linear	(128,)	Conv2d-5
9		Dense-3	LeakyReLU	(128,)	[Dense-1; Dense-2]
10	$p(\mathbf{X} \mid \mathbf{z}, s; \boldsymbol{\theta}_d)$	Concat1d-1	-	(138,)	[Dense-3; input_s_1d]
11		Reshape-1	-	(512,4,4)	Concat1d-1
12		ConvTr2d-1	LeakyReLU	(512,7,7)	Reshape-1
13		ConvTr2d-2	LeakyReLU	(256,13,13)	ConvTr2d-1
14		ConvTr2d-3	LeakyReLU	(128,25,25)	ConvTr2d-2
15		ConvTr2d-4	LeakyReLU	(64,50,50)	ConvTr2d-3
16		ConvTr2d-5	Tanh	(3,100,100)	ConvTr2d-4
17	$g(\mathbf{X}; \boldsymbol{\theta}_g)$	Conv2d-6	ReLU	(32,100,100)	input_x
18		Conv2d-7	ReLU	(32,50,50)	Conv2d-6
19		Conv2d-8	ReLU	(64,50,50)	Conv2d-7
20		Conv2d-9	ReLU	(64,25,25)	Conv2d-8
21		Conv2d-19	ReLU	(128,13,13)	Conv2d-9
22		Conv2d-11	ReLU	(128,13,13)	Conv2d-10
23	$h(\tilde{z}; \boldsymbol{\theta}_h)$	Dense-4	ReLU	(128,)	Conv2d-11
24		Dropout-1	-	(128,)	Dense-4
25		Dense-5	ReLU	(128,)	Dropout-1
26		Dropout-2	-	(128,)	Dense-5
27		Dense-6	Softmax	(10,)	Dropout-2

TABLE C.2: A detailed description of the architecture of the proposed DeSIRe model. The output shape is described as (#filters, rows, columns).

C.4 Visualization of the latent space

In this section, we present a 2-D visualization of the learned embedding via t-SNE, similarly to what we have done for our previous adversarial model (Section 4.3.4.4).

Figure C.2 depicts the t-SNE provided by the DeSIRe model and both implemented baselines in two test splits, of the MKLM dataset, with different degrees of inter-signer

variability. Figure C.3 illustrates the t-SNE plots obtained by the domain adaptation methods (DANN and DTML) for the same exact test splits. For easier comparison, we show the results for DeSIRe in both figures.

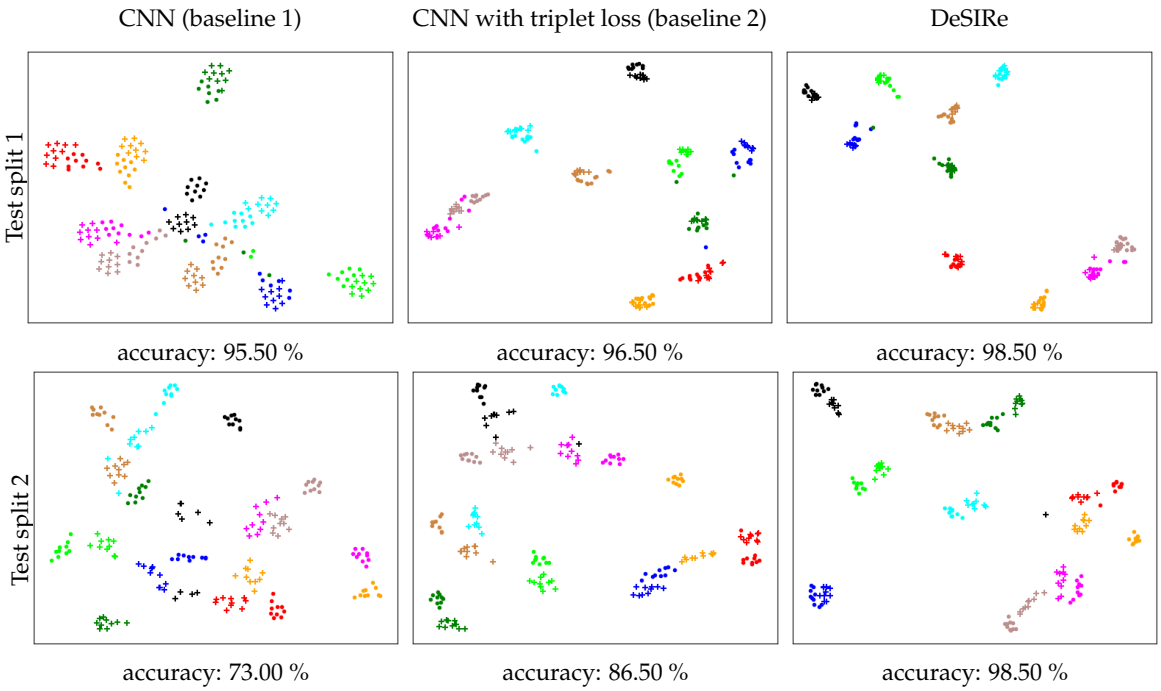


FIGURE C.2: Two-dimensional projection of the latent representation space provided by DeSIRe and both baselines, using t-SNE [182]. Markers • and + represent two different test signers from the MKLM dataset and the different colors correspond to the 10 sign classes. The accuracy of each model on each split is shown below each picture.

As it is possible to observe in Figures C.2 and C.3, all the implemented models achieved high classification accuracies on the test split 1. Nevertheless, the t-SNE plots clearly demonstrate the better capability of DeSIRe to impose signer-independence in the latent representations. The DeSIRe model yields a latent representation space in which latent representations of the same signer and different classes are close to each other and well mixed, while it keeps latent representations of different classes far apart.

The test split 2, depicted in the bottom row of Figures C.2 and C.3, is characterized by a larger inter-signer variability. Consequently, for this particular test split, the accuracy gains of DeSIRe are much more noticeable. The t-SNE plots support these classification results (see the bottom row of Figures C.2 and C.3). There, it is possible to observe that baseline 1 completely fails in the arrangement of the latent space. Specifically, the latent representations of different signers and the same class are too far apart. In addition, there is a clear overlap between clusters of different classes. Although the CNN with the triplet

loss (i.e. baseline 2) and both domain adaptation methods promoted slight improvements over the standard baseline CNN, the proposed DeSIRe model achieved by far the best inter-class separability.

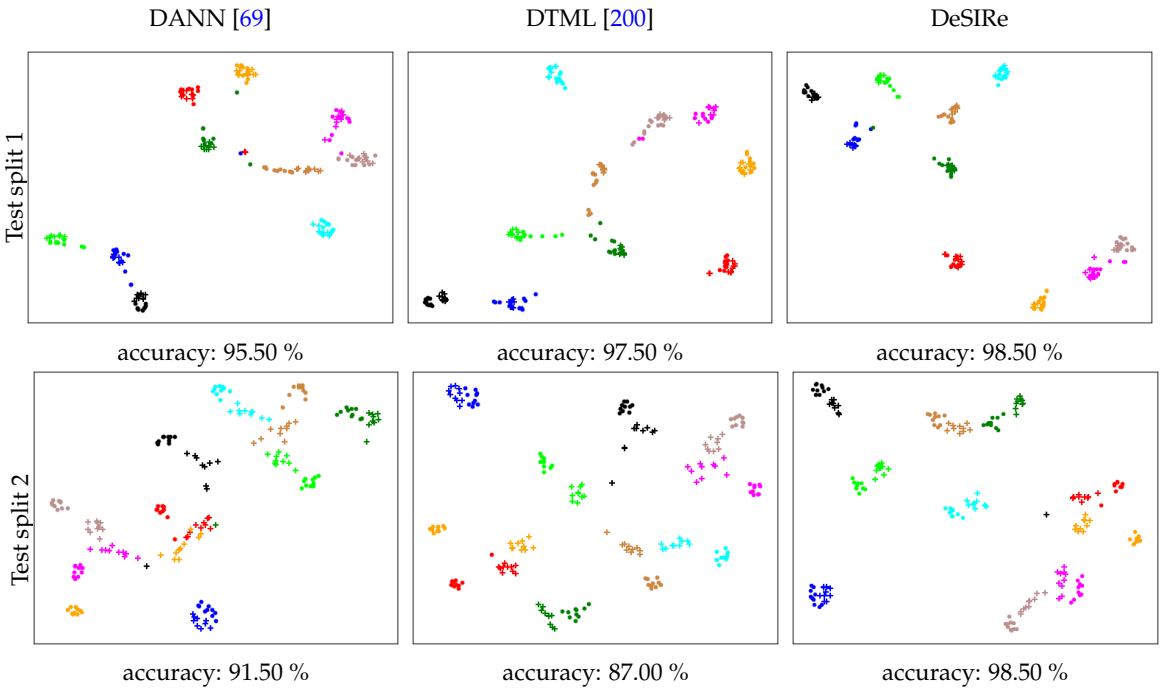


FIGURE C.3: Two-dimensional projection of the latent representation space provided by DeSIRe, DANN, and DTML, using t-SNE [182]. Markers • and + represent two different test signers from the MKLM dataset and the different colors correspond to the 10 sign classes. The accuracy of each model on each split is shown below each picture.

C.5 Cluster analysis in the latent space

In order to obtain an objective quality assessment of the produced latent representations, we have evaluated how well the model is able to cluster the different sign classes (and thus ignore the signer identity) in the latent space. For this purpose, we use two cluster validation metrics: the average Silhouette coefficient (Rousseeuw [202]) per cluster and the Dunn's index (Dunn [203]) per cluster.

The Silhouette coefficient for an observation i is computed as follows. Let C_i be the cluster (sign class) associated with the observation i. The average intra-cluster distance a_i and the minimum average inter-cluster distance b_i for the observation i are obtained as

follows:

$$a_i = \frac{1}{|C_i| - 1} \sum_{j \in C_i} d(i,j), \tag{C.1}$$

$$b_i = \min_{\overline{C} \neq C_i} \frac{1}{|\overline{C}|} \sum_{j \in \overline{C}} d(i,j), \tag{C.2}$$

where $|C_i|$ denotes the number of observations in the cluster $|C_i|$ and $d(i,j)$ is the Euclidean distance between the observations i and j. Then, the Silhouette index S_i for the observation i is defined as:

$$S_i = \frac{b_i - a_i}{\max(a_i, b_i)}. \tag{C.3}$$

Clearly, $-1 \leq S_i \leq 1$. Intuitively, clusters are desirably compact (small a_i) and well separated (large b_i), so a larger value of S_i indicates better clustering. However, this metric is defined per observation. Hence, in order to have a global measure of clustering quality, we compute the average Silhouette coefficient for each cluster.

Dunn's index follows a similar idea of measuring cluster compactness versus separation, but uses minimum and maximum distances instead of average distances, and is more sensitive to extreme and occasional errors. Specifically, Dunn's index D_C for a cluster C is defined as the ratio between the minimum inter-cluster distance δ_C from C to all other clusters (which measures cluster separation) and the maximum intra-cluster distance Δ_C for the cluster C (which measures cluster compactness):

$$\delta_C = \min_{i \in C, j \notin C} d(i,j), \quad \Delta_C = \max_{i,j \in C} d(i,j), \quad D_C = \frac{\delta_C}{\Delta_C}. \tag{C.4}$$

Again, according to this metric, larger values indicate better clustering. Results are shown in Table C.3. As anticipated by the analysis of the two-dimensional t-SNE projection in Figures C.2 and C.3, the results confirm that the DeSIRe model produces the most compact and separated sign clusters, when compared with the remaining models. This observation supports the signer-invariance property of the representations produced by the DeSIRe model: when exposed to images obtained from new signers, our model does a better job of grouping them according to the respective sign class only, ignoring the signer identity. Baseline 2 and DTML are also capable of producing fairly good sign clusters. This is not a surprising fact since both approaches include explicit penalties in the respective training objectives that favor compactness in the latent space among samples of the same sign class. The absence of such a compactness constraint in DANN allows its latent representations to be more widely spread over the latent space. As such, according to the adopted

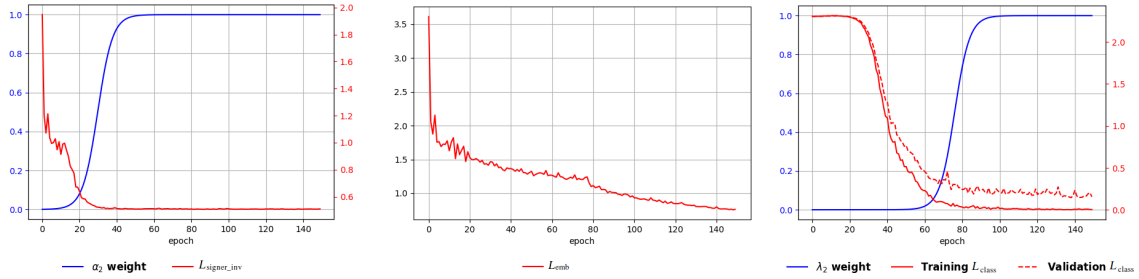
	Jochen-Triesch				MKLM				CorSiL			
	Dunn's		Silhouette		Dunn's		Silhouette		Dunn's		Silhouette	
	Average	min	Average	min	Average	min	Average	min	Average	min	Average	min
DANN [69]	0.165	0.105	0.457	0.342	0.380	0.102	0.531	0.253	0.310	0.205	0.281	0.109
DTML [200]	0.218	0.170	0.557	0.493	0.693	0.236	**0.653**	0.342	0.298	0.200	0.312	0.107
CNN (baseline 1)	0.171	0.132	0.405	0.326	0.378	0.159	0.537	0.295	0.346	0.184	0.316	0.112
CNN with triplet loss (baseline 2)	0.249	0.179	0.509	0.453	0.559	0.208	0.623	0.171	0.359	**0.210**	0.313	0.186
DeSIRe	**0.316**	**0.184**	**0.582**	**0.541**	**0.695**	**0.240**	0.646	**0.377**	**0.374**	0.186	**0.320**	**0.197**

TABLE C.3: Dunn's index and Silhouette coefficient for the sign class clusters in the latent space for the test data. These metrics were computed per cluster and the average and minimum results are reported for each model and dataset.

metrics, the obtained sign clusters are comparable to those obtained using a simple CNN (baseline 1), although the resulting sign classification accuracy is undoubtedly superior.

C.6 Unveiling the training behavior of DeSIRe

In this subsection, we further analyze the training process of the proposed DeSIRe model. Figure C.4 shows the behavior of different loss terms, over 150 epochs of training on the MKLM dataset, with the sigmoid annealing schedules in place. Specifically, we have plotted the curves of the key loss terms, $L_{\text{signer\_inv}}$ and L_{emb}, responsible for promoting signer-invariant latent representations (see Figures C.4a and C.4b, respectively). In addition, we have also plotted the curve of the classification loss term L_{class}, which trains the model to predict the output sign labels.



(A) (B) (C)

FIGURE C.4: Training behavior of the proposed DeSIRe model: (A) evolution of the $L_{\text{signer\_inv}}$ loss term alongside the corresponding weight α_2 according to a sigmoid annealing schedule; (B) evolution of the L_{emb} term value; and (C) evolution of training and validation L_{class} curves alongside the corresponding weight λ_2 according to a sigmoid annealing schedule.

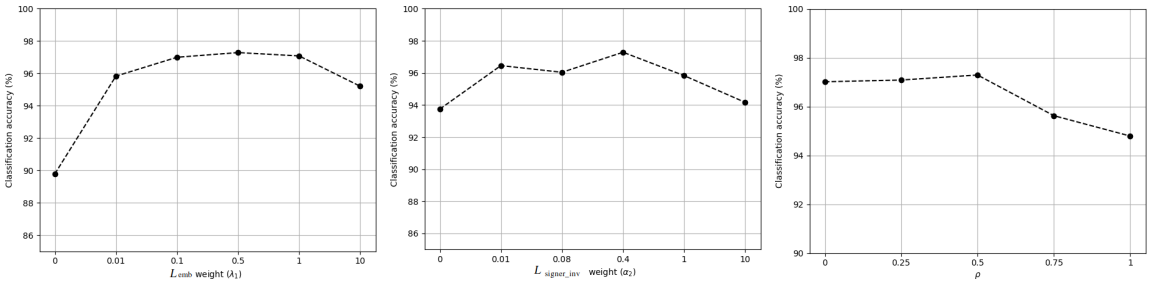
Figure C.4c depicts the observed behavior for the classification loss term L_{class}. As previously explained in Section C.2, at the start of training, the classification weight λ_2 is set to zero and the L_{emb} error is backpropagated only through the convolutional block of the classifier module. Therefore, in the first training epochs, L_{class} remains at a high value, as the classifier predicts random guesses (see Figure C.4c). Then, L_{class} drops quickly once

the classification weight λ_2 starts increasing. This shows that the feature representations learned in the previous phase are highly discriminative for the classification task. On the other hand, the CVAE module starts to be trained on the reconstruction task as soon as the training process begins. At this stage, the sampling scheme associated with the reconstruction loss L_{rec} is the only mechanism promoting signer-invariance. After a few epochs, the KL weights α_1 and α_2 start increasing and the CVAE is further enforced to produce signer-invariant latent representations. In particular, Figure C.4a shows the evolution of the signer-invariance loss $L_{\text{signer\_inv}}$ together with the respective weight α_2 and attests that, at the end of training, the encoder produces signer-independent embeddings in the training data. Finally, Figure C.4b depicts a consistent decrease of the embedding loss L_{emb} throughout the entire training routine.

C.7 Hyperparameter sensitivity analysis

This subsection presents a sensitivity analysis of three key hyperparameters of the proposed DeSIRe model, namely the L_{emb} weight λ_1, the signer-invariance weight α_2 and the probability of changing the signer identity fed to the decoder network ρ. For this purpose, we plotted the curves of the average test accuracy of the proposed model with varying values of $\lambda_1 \in [0, 10]$, $\alpha_2 \in [0, 10]$, and $\rho \in [0, 1]$ (Figure C.5). Some interesting conclusions can be drawn from these plots. Particularly, when $\alpha_1 = 0$, L_{class} is the only loss term still active. Accordingly, the proposed DeSIRe model has exactly the same behavior as baseline 1, which results in a significant drop in the test accuracy. When $\lambda_1 \neq 0$ and $\alpha_2 = 0$, the loss terms L_{emb}, L_{rec} and L_{prior} become active and only $L_{\text{signer\_inv}}$ is inactive. Under this setting, the test accuracy increases to nearly 94% (see Figure C.5b). Here, the classifier is trained to follow the latent representations produced by the CVAE. Although the term $L_{\text{signer\_inv}}$ is not present, signer-invariance is still promoted by (i) the L_{prior} loss term; and (ii) by conditioning the decoder on the signer identity, which is drawn from a random distribution. Finally, when λ_1 and α_2 are both set to their optimal values, all loss terms are active and a maximum test accuracy is achieved. The observed accuracy gain clearly supports the beneficial regularizing effect of the $L_{\text{signer\_inv}}$ loss term, which explicitly promotes signer-invariant representations.

In addition, it is worth mentioning that the proposed DeSIRe model is quite robust to these hyperparameters as the accuracy curves remain quite stable over a large range of values (i.e. $\lambda_1, \alpha_2 \in [0.01, 1]$). The impact of ρ, which controls the proposed sampling



(A) (B) (C)

FIGURE C.5: Hyperparameter sensitivity analysis: (A) DeSIRe accuracy on the Jochen-Triesch dataset with varying values of $\lambda_1 \in [0, 10]$ while $\alpha_2 = 0.4$ and $\rho = 0.5$; (B) DeSIRe accuracy on the Jochen-Triesch dataset with varying values of $\alpha_2 \in [0, 10]$ while $\lambda_1 = 0.5$ and $\rho = 0.5$; and (C) DeSIRe accuracy on the Jochen-Triesch dataset with varying values of $\rho \in [0, 1]$ while $\lambda_1 = 0.5$ and $\alpha_2 = 0.4$.

scheme of the signer identity in the learning process, is depicted in Figure C.5c. From this figure, it is possible to observe that ρ should be set around 0.5. The test accuracy progressively decreases when ρ falls into the interval $[0.75, 1]$. In these cases, the decoder will be trained most of the time to reconstruct an image of a different signer than the one that was used to produce the encoding. This naturally makes the reconstruction task and the overall CVAE training process too difficult, explaining the noticeable significant performance drop.

Bibliography

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

[2] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009. [Cited on pages 6, 8, and 10.]

[3] W. Turin, *Continuous State HMM*. Boston, MA: Springer US, 2004, pp. 295–340. [Cited on page 6.]

[4] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, August 2006. [Cited on page 7.]

[5] L. R. Rabiner and B.-H. Juang, "An introduction to hidden Markov models," *ieee assp magazine*, vol. 3, no. 1, pp. 4–16, 1986. [Cited on page 7.]

[6] L. Baum, "An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process," *Inequalities*, vol. 3, pp. 1–8, 1972. [Cited on pages 11 and 21.]

[7] J. J. Sakurai and J. Napolitano, *Modern Quantum Mechanics*, 2nd ed. Cambridge University Press, 2017. [Cited on page 12.]

[8] J. Rustagi, *Variational Methods in Statistics*, ser. Mathematics in science and engineering: a series of monographs and textbooks. Academic Press, 1976, no. vol. 121. [Cited on page 12.]

[9] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, cite arxiv:1312.6114. [Cited on pages 12 and 54.]

[10] A. Allahdadi, D. Pernes, J. S. Cardoso, and R. Morla, "Hidden Markov models on a self-organizing map for anomaly detection in 802.11 wireless networks," *Neural Computing and Applications*, pp. 1–18, 2021. [Cited on pages 17 and 34.]

[11] D. Pernes and J. S. Cardoso, "SpaMHMM: Sparse mixture of hidden Markov models for graph connected entities," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–10. [Cited on page 17.]

[12] F. De Vico Fallani, J. Richiardi, M. Chavez, and S. Achard, "Graph analysis of functional brain networks: practical issues in translational neuroscience," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 369, no. 1653, 2014. [Cited on page 17.]

[13] M. R. Tora, J. Chen, and J. J. Little, "Classification of puck possession events in ice hockey," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, July 2017, pp. 147–154. [Cited on page 18.]

[14] R. Theagarajan, F. Pala, X. Zhang, and B. Bhanu, "Soccer: Who has the ball? generating visual analytics and player statistics," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018. [Cited on page 18.]

[15] N. Cheng, F. Lyu, J. Chen, W. Xu, H. Zhou, S. Zhang, and X. S. Shen, "Big data driven vehicular networks," *IEEE Network*, vol. 32, no. 6, pp. 160–167, November 2018. [Cited on page 18.]

[16] J. Gama and M. M. Gaber, *Learning from data streams: processing techniques in sensor networks*. Springer, 2007. [Cited on page 18.]

[17] S. Laxman, V. Tankasali, and R. W. White, "Stream prediction using a generative model based on frequent episodes in event sequences," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 08 2008, pp. 453–461. [Cited on page 18.]

[18] M. Z. Hayat and M. R. Hashemi, "A dct based approach for detecting novelty and concept drift in data streams," in *2010 International Conference of Soft Computing and Pattern Recognition*, Dec 2010, pp. 373–378. [Cited on page 18.]

[19] A. Hofmann and B. Sick, "Online intrusion alert aggregation with generative data stream modeling," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 2, pp. 282–294, 2011. [Cited on page 18.]

[20] D. Baron, M. F. Duarte, M. B. Wakin, S. Sarvotham, and R. G. Baraniuk, "Distributed compressive sensing," *CoRR*, vol. abs/0901.3403, 2009. [Cited on page 19.]

[21] A. Allahdadi, R. Morla, and J. S. Cardoso, "802.11 wireless simulation and anomaly detection using HMM and UBM," *SIMULATION*, vol. 96, no. 12, pp. 939–956, 2020. [Cited on pages 20, 23, 32, 33, and 43.]

[22] P. Somervuo, "Competing hidden Markov models on the self-organizing map," in *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, vol. 3. IEEE, 2000, pp. 169–174. [Cited on page 20.]

[23] M. Kurimo and P. Somervuo, "Using the self-organizing map to speed up the probability density estimation for speech recognition with mixture density HMMs," in *Spoken Language, 1996. ICSLP 96. Proceedings, Fourth International Conference on*, vol. 1. IEEE, 1996, pp. 358–361. [Cited on page 20.]

[24] H. Morimoto, "Hidden Markov models and self-organizing maps applied to stroke incidence," *Open Journal of Applied Sciences*, vol. 6, no. 3, pp. 158–168, 2016. [Cited on pages 20 and 22.]

[25] C. Ferles and A. Stafylopatis, "Self-organizing hidden Markov model map (SOHMMM)," *Neural Networks*, vol. 48, pp. 133 – 147, 2013. [Cited on pages 21 and 22.]

[26] C. Ferles, G. Siolas, and A. Stafylopatis, "Scaled self-organizing map–hidden Markov model architecture for biological sequence clustering," *Applied Artificial Intelligence*, vol. 27, no. 6, pp. 461–495, 2013. [Cited on page 21.]

[27] M. Lebbah, R. Jaziri, Y. Bennani, and J.-H. Chenot, "Probabilistic self-organizing map for clustering and visualizing non-IID data," *International Journal of Computational Intelligence and Applications*, vol. 14, no. 02, p. 1550007, 2015. [Cited on page 21.]

[28] P. Baldi and Y. Chauvin, "Smooth on-line learning algorithms for hidden Markov models," *Neural Computation*, vol. 6, no. 2, pp. 307–318, 1994. [Cited on pages 21 and 54.]

[29] G. Niina and H. Dozono, "The spherical hidden Markov self organizing map for learning time series data," in *International Conference on Artificial Neural Networks*. Springer, 2012, pp. 563–570. [Cited on page 21.]

[30] N. Yamaguchi, "Self-organizing hidden Markov models," in *International Conference on Neural Information Processing*. Springer, 2010, pp. 454–461. [Cited on page 21.]

[31] G. Caridakis, K. Karpouzis, A. Drosopoulos, and S. Kollias, "SOMM: Self organizing Markov map for gesture recognition," *Pattern Recognition Letters*, vol. 31, no. 1, pp. 52–59, 2010. [Cited on page 21.]

[32] R. Jaziri, M. Lebbah, Y. Bennani, and J.-H. Chenot, "SOS-HMM: self-organizing structure of hidden Markov model," in *International Conference on Artificial Neural Networks*. Springer, 2011, pp. 87–94. [Cited on page 21.]

[33] C. Ferles and A. Stafylopatis, "Sequence clustering with the self-organizing hidden Markov model map," in *2008 8th IEEE International Conference on BioInformatics and BioEngineering*. IEEE, 2008, pp. 1–7. [Cited on pages 22, 23, 24, and 25.]

[34] C. Ferles, W.-S. Beaufort, and V. Ferle, "Self-organizing hidden Markov model map (SOHMMM): Biological sequence clustering and cluster visualization," in *Hidden Markov Models*. Springer, 2017, pp. 83–101. [Cited on page 22.]

[35] W. Khreich, E. Granger, A. Miri, and R. Sabourin, "A survey of techniques for incremental learning of HMM parameters," *Information Sciences*, vol. 197, pp. 105–130, 2012. [Cited on pages 22 and 54.]

[36] S.-B. Cho, "Incorporating soft computing techniques into a probabilistic intrusion detection system," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 32, no. 2, pp. 154–160, 2002. [Cited on pages 22 and 23.]

[37] W. Wang, X. Guan, X. Zhang, and L. Yang, "Profiling program behavior for anomaly intrusion detection based on the transition and frequency property of computer audit data," *Computers & Security*, vol. 25, no. 7, pp. 539–550, 2006. [Cited on pages 22 and 23.]

[38] A. Allahdadi, R. Morla, and J. S. Cardoso, "Outlier detection in 802.11 wireless access points using hidden Markov models," in *Wireless and Mobile Networking Conference (WMNC), 2014 7th IFIP*. IEEE, 2014, pp. 1–8. [Cited on page 23.]

[39] A. Allahdadi and R. Morla, "Anomaly detection and modeling in 802.11 wireless networks," *Journal of Network and Systems Management*, vol. 27, no. 1, pp. 3–38, Jan 2019. [Cited on pages 23 and 32.]

[40] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989. [Cited on page 29.]

[41] B.-H. Juang and L. R. Rabiner, "A probabilistic distance measure for hidden Markov models," *AT&T Technical Journal*, vol. 64, no. 2, pp. 391–408, 1985. [Cited on page 29.]

[42] OMNeT++, discrete event simulator. https://www.omnetpp.org. [Cited on pages 31 and 43.]

[43] Inet framework. https://inet.omnetpp.org. [Cited on pages 31 and 43.]

[44] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977. [Cited on page 37.]

[45] Z. Yang, J. Zhao, B. Dhingra, K. He, W. W. Cohen, R. Salakhutdinov, and Y. LeCun, "Glomo: Unsupervisedly learned relational graphs as transferable representations," 2018. [Cited on page 40.]

[46] S. Lebedev. hmmlearn, hidden Markov models in python, with scikit-learn like API. https://github.com/hmmlearn/hmmlearn. [Cited on page 40.]

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. [Cited on pages 42 and 76.]

[48] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik, "Recurrent network models for human dynamics," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4346–4354. [Cited on pages 46, 47, and 48.]

[49] J. Martinez, M. J. Black, and J. Romero, "On human motion prediction using recurrent neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 4674–4683. [Cited on pages 46 and 48.]

[50] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-rnn: Deep learning on spatio-temporal graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5308–5317. [Cited on pages 46 and 48.]

[51] D. Pavllo, D. Grangier, and M. Auli, "Quaternet: A quaternion-based recurrent model for human motion," *arXiv preprint arXiv:1805.06485*, 2018. [Cited on pages 46 and 48.]

[52] C. Ionescu, F. Li, and C. Sminchisescu, "Latent structured models for human pose estimation," in *International Conference on Computer Vision*, 2011. [Cited on page 46.]

[53] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014. [Cited on page 46.]

[54] C. M. Bishop, "Mixture density networks," Citeseer, Tech. Rep., 1994. [Cited on page 52.]

[55] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[56] L. Bazzani, H. Larochelle, and L. Torresani, "Recurrent mixture density network for spatiotemporal visual attention," *arXiv preprint arXiv:1603.08199*, 2016. [Cited on page 52.]

[57] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. Jan, pp. 993–1022, 2003. [Cited on page 54.]

[58] V. V. Digalakis, "Online adaptation of hidden Markov models using incremental estimation algorithms," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 253–261, 1999. [Cited on page 54.]

[59] G. Mongillo and S. Deneve, "Online learning with hidden Markov models," *Neural computation*, vol. 20, pp. 1706–16, 08 2008. [Cited on page 54.]

[60] T. Rydén, "On recursive estimation for hidden Markov models," *Stochastic Processes and their Applications*, vol. 66, no. 1, pp. 79–96, 1997. [Cited on page 54.]

[61] F. T. de Andrade, "Adversarial domain adaptation for sensor networks," Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2020. [Cited on pages xvii, xviii, 57, 72, 73, 74, and 82.]

[62] D. Pernes and J. S. Cardoso, "Tackling unsupervised multi-source domain adaptation with optimism and consistency," *CoRR*, vol. abs/2009.13939, 2020. [Cited on page 57.]

[63] H. Daumé III, A. Kumar, and A. Saha, "Frustratingly easy semi-supervised domain adaptation," in *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*. Association for Computational Linguistics, 2010, pp. 53–59. [Cited on page 58.]

[64] J. Donahue, J. Hoffman, E. Rodner, K. Saenko, and T. Darrell, "Semi-supervised domain adaptation with instance constraints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 668–675. [Cited on page 58.]

[65] A. Kumar, A. Saha, and H. Daume, "Co-regularization based semi-supervised domain adaptation," in *Advances in Neural Information Processing Systems*, 2010, pp. 478–486. [Cited on page 58.]

[66] K. Saito, D. Kim, S. Sclaroff, T. Darrell, and K. Saenko, "Semi-supervised domain adaptation via minimax entropy," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 8050–8058. [Cited on page 58.]

[67] T. Yao, Y. Pan, C.-W. Ngo, H. Li, and T. Mei, "Semi-supervised domain adaptation with subspace learning for visual recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2142–2150. [Cited on page 58.]

[68] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann, "Unsupervised domain adaptation by domain invariant projection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 769–776. [Cited on page 58.]

[69] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International Conference on Machine Learning*, 2015, pp. 1180–1189. [Cited on pages 58, 66, 68, 70, 79, 83, 91, 93, 99, 102, 103, 108, 126, 127, 128, 155, and 157.]

[70] G. Kang, L. Jiang, Y. Yang, and A. G. Hauptmann, "Contrastive adaptation network for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4893–4902. [Cited on page 58.]

[71] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Unsupervised domain adaptation with residual transfer networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 136–144. [Cited on page 58.]

[72] H. Zhao, S. Zhang, G. Wu, J. M. Moura, J. P. Costeira, and G. J. Gordon, "Adversarial multiple source domain adaptation," in *Advances in Neural Information Processing*

Systems, 2018, pp. 8559–8570. [Cited on pages 58, 59, 61, 66, 68, 85, 86, 88, 90, 91, 93, 99, and 141.]

[73] 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, no. 1-2, pp. 151–175, 2010. [Cited on pages 58, 59, 60, 61, 84, and 137.]

[74] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, "Analysis of representations for domain adaptation," in *Advances in Neural Information Processing Systems*, 2007, pp. 137–144. [Cited on page 58.]

[75] J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman, "Learning bounds for domain adaptation," in *Advances in Neural Information Processing Systems*, 2008, pp. 129–136. [Cited on pages 58 and 137.]

[76] C. Cortes and M. Mohri, "Domain adaptation and sample bias correction theory and algorithm for regression," *Theoretical Computer Science*, vol. 519, pp. 103–126, 2014. [Cited on page 58.]

[77] R. Gopalan, R. Li, and R. Chellappa, "Unsupervised adaptation across domain shifts by generating intermediate data representations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2288–2302, 2013. [Cited on page 58.]

[78] J. Hoffman, M. Mohri, and N. Zhang, "Algorithms and theory for multiple-source adaptation," in *Advances in Neural Information Processing Systems*, 2018, pp. 8246–8256. [Cited on page 58.]

[79] H. Zhao, R. T. Des Combes, K. Zhang, and G. Gordon, "On learning invariant representations for domain adaptation," in *International Conference on Machine Learning*, 2019, pp. 7523–7532. [Cited on pages 58, 83, 85, 86, 138, and 141.]

[80] H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, and M. Marchand, "Domain-adversarial neural networks," *arXiv preprint arXiv:1412.4446*, 2014. [Cited on page 58.]

[81] C. J. Becker, C. M. Christoudias, and P. Fua, "Non-linear domain adaptation with boosting," in *Advances in Neural Information Processing Systems*, 2013, pp. 485–493. [Cited on page 58.]

[82] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2960–2967. [Cited on page 58.]

[83] I.-H. Jhuo, D. Liu, D. Lee, and S.-F. Chang, "Robust visual domain adaptation with low-rank reconstruction," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 2168–2175. [Cited on page 58.]

[84] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," *arXiv preprint arXiv:1502.02791*, 2015. [Cited on pages 58 and 66.]

[85] C. Louizos, K. Swersky, Y. Li, M. Welling, and R. Zemel, "The variational fair autoencoder," in *International Conference on Learning Representations*, 2016. [Cited on page 58.]

[86] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. [Cited on page 58.]

[87] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7167–7176. [Cited on page 58.]

[88] 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)*, 2017, pp. 643–653. [Cited on page 58.]

[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*, 2018, pp. 4694–4703. [Cited on pages 58, 66, 83, 91, and 93.]

[90] Y. Mansour, M. Mohri, and A. Rostamizadeh, "Domain adaptation: Learning bounds and algorithms," in *22nd Conference on Learning Theory, COLT 2009*, 2009. [Cited on page 58.]

[91] A. Schoenauer-Sebag, L. Heinrich, M. Schoenauer, M. Sebag, L. Wu, and S. Altschuler, "Multi-domain adversarial learning," in *International Conference on Learning Representations*, 2019. [Cited on pages 58, 66, 83, and 87.]

[92] K. Zhang, M. Gong, and B. Schölkopf, "Multi-source domain adaptation: A causal view," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015. [Cited on page 58.]

[93] C. Cortes and M. Mohri, "Domain adaptation in regression," in *Algorithmic Learning Theory*, J. Kivinen, C. Szepesvári, E. Ukkonen, and T. Zeugmann, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 308–323. [Cited on page 59.]

[94] Z. Lipton, Y.-X. Wang, and A. Smola, "Detecting and correcting for label shift with black box predictors," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 3122–3130. [Cited on page 63.]

[95] 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*, 2014, pp. 530–538. [Cited on page 63.]

[96] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang, "Domain adaptation under target and conditional shift," in *International Conference on Machine Learning*, 2013, pp. 819–827. [Cited on pages 63 and 64.]

[97] Y. S. Chan and H. T. Ng, "Word sense disambiguation with distribution estimation," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, ser. IJCAI'05. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005, p. 1010–1015. [Cited on page 63.]

[98] A. Storkey, "When training and test sets are different: characterizing learning transfer," *Dataset shift in machine learning*, pp. 3–28, 2009. [Cited on page 63.]

[99] G. I. Webb, L. K. Lee, B. Goethals, and F. Petitjean, "Analyzing concept drift and shift from sample data," *Data Mining and Knowledge Discovery*, vol. 32, no. 5, pp. 1179–1199, 2018. [Cited on page 64.]

[100] J. Gama, I. Žliobaitundefined, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, Mar. 2014. [Cited on page 64.]

[101] M. Sugiyama, M. Krauledat, and K.-R. Müller, "Covariate shift adaptation by importance weighted cross validation." *Journal of Machine Learning Research*, vol. 8, pp. 985–1005, 05 2007. [Cited on page 65.]

[102] 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. [Cited on page 65.]

[103] C. Cortes, Y. Mansour, and M. Mohri, "Learning bounds for importance weighting," in *Advances in Neural Information Processing Systems*, 2010, pp. 442–450. [Cited on page 65.]

[104] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy, "Optimal transport for domain adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 9, pp. 1853–1865, 2017. [Cited on page 65.]

[105] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy, "Joint distribution optimal transportation for domain adaptation," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Cited on page 65.]

[106] R. Turrisi, R. Flamary, A. Rakotomamonjy, and M. Pontil, "Multi-source domain adaptation via weighted joint distributions optimal transport," *arXiv preprint arXiv:2006.12938*, 2020. [Cited on page 65.]

[107] Z. Pei, Z. Cao, M. Long, and J. Wang, "Multi-adversarial domain adaptation," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. [Cited on pages 66, 83, 91, and 92.]

[108] E. Bareinboim and J. Pearl, "Causal inference and the data-fusion problem," *Proceedings of the National Academy of Sciences*, vol. 113, no. 27, pp. 7345–7352, 2016. [Cited on page 67.]

[109] M. Rojas-Carulla, B. Schölkopf, R. Turner, and J. Peters, "Invariant models for causal transfer learning," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 1309–1342, 2018. [Cited on page 67.]

[110] S. Magliacane, T. van Ommen, T. Claassen, S. Bongers, P. Versteeg, and J. M. Mooij, "Domain adaptation by using causal inference to predict invariant conditional distributions," in *Advances in Neural Information Processing Systems*, 2018. [Cited on page 67.]

[111] J. Peters, J. M. Mooij, D. Janzing, and B. Schölkopf, "Causal discovery with continuous additive noise models," *Journal of Machine Learning Research*, vol. 15, pp. 2009–2053, 2014. [Cited on page 67.]

[112] B. Schölkopf, F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio, "Toward causal representation learning," *Proceedings of the IEEE*, 2021. [Cited on page 67.]

[113] M.-H. Chen, Z. Kira, and G. AlRegib, "Temporal attentive alignment for video domain adaptation," in *International Conference on Computer Vision (ICCV)*, 2019. [Cited on page 68.]

[114] 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. [Cited on page 68.]

[115] 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. [Cited on pages xvii, 68, 70, and 71.]

[116] A. B. Chan and N. Vasconcelos, "Modeling, clustering, and segmenting video with mixtures of dynamic textures," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 5, pp. 909–926, 2008. [Cited on page 76.]

[117] S. Zhang, G. Wu, J. P. Costeira, and J. M. Moura, "Understanding traffic density from large-scale web camera data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5898–5907. [Cited on page 77.]

[118] P. M. Ferreira, D. Pernes, A. Rebelo, and J. S. Cardoso, "DeSIRe: Deep signer-invariant representations for sign language recognition," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019. [Cited on pages 83 and 97.]

[119] K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," *arXiv preprint arXiv:2001.07685*, 2020. [Cited on pages 89 and 90.]

[120] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical data augmentation with no separate search," *arXiv preprint arXiv:1909.13719*, 2019. [Cited on page 90.]

[121] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014. [Cited on page 90.]

[122] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1406–1415. [Cited on pages 91, 92, and 93.]

[123] 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. [Cited on page 91.]

[124] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Advances in Neural Information Processing Systems*, 2011. [Cited on page 91.]

[125] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *European Conference on Computer Vision*. Springer, 2010, pp. 213–226. [Cited on page 92.]

[126] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778. [Cited on page 92.]

[127] J. Blitzer, M. Dredze, and F. Pereira, "Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification," in *Proceedings of the 45th annual meeting of the association of computational linguistics*, 2007, pp. 440–447. [Cited on page 92.]

[128] M. Chen, Z. Xu, K. Weinberger, and F. Sha, "Marginalized denoising autoencoders for domain adaptation," *arXiv preprint arXiv:1206.4683*, 2012. [Cited on page 92.]

[129] P. M. Ferreira, D. Pernes, A. Rebelo, and J. S. Cardoso, "Learning signer-invariant representations with adversarial training," in *Twelfth International Conference on Machine Vision (ICMV 2019)*, vol. 11433. International Society for Optics and Photonics, 2020, p. 114333D. [Cited on page 97.]

[130] ——, "Signer-independent sign language recognition with adversarial neural networks," *International Journal of Machine Learning and Computing*, vol. 11, no. 2, 2021. [Cited on page 97.]

[131] P. M. Ferreira, A. F. Sequeira, D. Pernes, A. Rebelo, and J. S. Cardoso, "Adversarial learning for a robust iris presentation attack detection method against unseen attack presentations," in *2019 International Conference of the Biometrics Special Interest Group (BIOSIG)*, 2019, pp. 1–7. [Cited on pages 97 and 119.]

[132] G. Blanchard, G. Lee, and C. Scott, "Generalizing from several related classification tasks to a new unlabeled sample," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011. [Cited on page 98.]

[133] K. Muandet, D. Balduzzi, and B. Schölkopf, "Domain generalization via invariant feature representation," in *International Conference on Machine Learning*. PMLR, 2013, pp. 10–18. [Cited on page 98.]

[134] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy, "Domain generalization: A survey," *arXiv preprint arXiv:2103.02503*, 2021. [Cited on pages 99 and 100.]

[135] I. Albuquerque, J. Monteiro, M. Darvishi, T. H. Falk, and I. Mitliagkas, "Generalizing to unseen domains via distribution matching," *arXiv preprint arXiv:1911.00804*, 2019. [Cited on page 99.]

[136] H. Li, S. J. Pan, S. Wang, and A. C. Kot, "Domain generalization with adversarial feature learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5400–5409. [Cited on page 99.]

[137] M. Ghifary, W. B. Kleijn, M. Zhang, and D. Balduzzi, "Domain generalization for object recognition with multi-task autoencoders," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2551–2559. [Cited on page 99.]

[138] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto, "Unified deep supervised domain adaptation and generalization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5715–5725. [Cited on page 99.]

[139] S. Aslani, V. Murino, M. Dayan, R. Tam, D. Sona, and G. Hamarneh, "Scanner invariant multiple sclerosis lesion segmentation from mri," in *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*. IEEE, 2020, pp. 781–785. [Cited on page 99.]

[140] T. Matsuura and T. Harada, "Domain generalization using a mixture of multiple latent domains," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11 749–11 756. [Cited on page 99.]

[141] R. Shao, X. Lan, J. Li, and P. C. Yuen, "Multi-adversarial discriminative deep domain generalization for face presentation attack detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 023–10 031. [Cited on page 99.]

[142] Y. Li, X. Tian, M. Gong, Y. Liu, T. Liu, K. Zhang, and D. Tao, "Deep domain generalization via conditional invariant adversarial networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 624–639. [Cited on page 99.]

[143] Z. Xu, W. Li, L. Niu, and D. Xu, "Exploiting low-rank structure from latent domains for domain generalization," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 628–643. [Cited on page 99.]

[144] A. D'Innocente and B. Caputo, "Domain generalization with domain-specific aggregation modules," in *German Conference on Pattern Recognition*. Springer, 2018, pp. 187–198. [Cited on page 100.]

[145] K. Zhou, Y. Yang, Y. Qiao, and T. Xiang, "Domain adaptive ensemble learning," *arXiv preprint arXiv:2003.07325*, 2020. [Cited on page 100.]

[146] S. Wang, L. Yu, K. Li, X. Yang, C.-W. Fu, and P.-A. Heng, "Dofe: Domain-oriented feature embedding for generalizable fundus image segmentation on unseen datasets," *IEEE Transactions on Medical Imaging*, vol. 39, no. 12, pp. 4237–4248, 2020. [Cited on page 100.]

[147] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *European conference on computer vision*. Springer, 2016, pp. 649–666. [Cited on page 100.]

[148] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1422–1430. [Cited on page 100.]

[149] D. Wei, J. J. Lim, A. Zisserman, and W. T. Freeman, "Learning and using the arrow of time," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8052–8060. [Cited on page 100.]

[150] F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi, "Domain generalization by solving jigsaw puzzles," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2229–2238. [Cited on page 100.]

[151] S. Wang, L. Yu, C. Li, C.-W. Fu, and P.-A. Heng, "Learning from extrinsic and intrinsic supervisions for domain generalization," in *European Conference on Computer Vision*. Springer, 2020, pp. 159–176. [Cited on page 100.]

[152] J. Wang, C. Lan, C. Liu, Y. Ouyang, and T. Qin, "Generalizing to unseen domains: A survey on domain generalization," *arXiv preprint arXiv:2103.03097*, 2021. [Cited on page 100.]

[153] M. Ebrahim Al-Ahdal and M. T. Nooritawati, "Review in sign language recognition systems," in *2012 IEEE Symposium on Computers Informatics (ISCI)*, March 2012, pp. 52–57. [Cited on page 101.]

[154] L. Pigou, S. Dieleman, P.-J. Kindermans, and B. Schrauwen, "Sign language recognition using convolutional neural networks," in *Computer Vision - ECCV 2014 Workshops*, L. Agapito, M. M. Bronstein, and C. Rother, Eds. Cham: Springer International Publishing, 2015, pp. 572–578. [Cited on pages 101 and 105.]

[155] O. Koller, H. Ney, and R. Bowden, "Deep hand: How to train a cnn on 1 million hand images when your data is continuous and weakly labelled," in *2016 IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), June 2016, pp. 3793–3802. [Cited on pages 101 and 105.]

[156] D. Wu, L. Pigou, P. Kindermans, N. D. Le, L. Shao, J. Dambre, and J. Odobez, "Deep dynamic neural networks for multimodal gesture segmentation and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 8, pp. 1583–1597, Aug 2016. [Cited on pages 101 and 105.]

[157] N. Neverova, C. Wolf, G. Taylor, and F. Nebout, "Moddrop: Adaptive multi-modal gesture recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 8, pp. 1692–1706, Aug 2016. [Cited on pages 101 and 105.]

[158] P. Kumar, H. Gauba, P. P. Roy, and D. P. Dogra, "A multimodal framework for sensor based sign language recognition," *Neurocomputing*, vol. 259, pp. 21 – 38, 2017, multimodal Media Data Understanding and Analytics. [Cited on pages 101 and 105.]

[159] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014. [Cited on page 102.]

[160] C. Feutry, P. Piantanida, Y. Bengio, and P. Duhamel, "Learning anonymized representations with adversarial neural networks," *arXiv preprint arXiv:1802.09386*, 2018. [Cited on pages 102, 103, and 108.]

[161] G. Marin, F. Dominio, and P. Zanuttigh, "Hand gesture recognition with leap motion and kinect devices," in *2014 IEEE International conference on image processing (ICIP)*. IEEE, 2014, pp. 1565–1569. [Cited on pages 103, 110, 112, and 113.]

[162] ——, "Hand gesture recognition with jointly calibrated leap motion and depth sensor," *Multimedia Tools and Applications*, vol. 75, no. 22, pp. 14 991–15 015, 2016. [Cited on pages 103 and 110.]

[163] D. Guo, W. Zhou, H. Li, and M. Wang, "Online early-late fusion based on adaptive hmm for sign language recognition," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 14, no. 1, pp. 8:1–8:18, Dec. 2017. [Cited on page 103.]

[164] ——, "Hierarchical lstm for sign language translation," in *AAAI*, 2018. [Cited on page 103.]

[165] S. Wang, D. Guo, W.-g. Zhou, Z.-J. Zha, and M. Wang, "Connectionist temporal fusion for sign language translation," in *Proceedings of the 26th ACM International Conference on Multimedia*, ser. MM '18. New York, NY, USA: ACM, 2018, pp. 1483–1491. [Cited on page 103.]

[166] U. von Agris, D. Schneider, J. Zieren, and K. Kraiss, "Rapid signer adaptation for isolated sign language recognition," in *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, June 2006, pp. 159–159. [Cited on page 104.]

[167] U. von Agris, C. Blomer, and K. Kraiss, "Rapid signer adaptation for continuous sign language recognition using a combined approach of eigenvoices, mllr, and map," in *2008 19th International Conference on Pattern Recognition*, Dec 2008, pp. 1–4. [Cited on page 104.]

[168] R. Kuhn, J. . Junqua, P. Nguyen, and N. Niedzielski, "Rapid speaker adaptation in eigenvoice space," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 6, pp. 695–707, Nov 2000. [Cited on page 104.]

[169] T. Kim, W. Wang, H. Tang, and K. Livescu, "Signer-independent fingerspelling recognition with deep neural network adaptation," *CoRR*, vol. abs/1602.04278, 2016. [Cited on page 104.]

[170] F. Yin, X. Chai, Y. Zhou, and X. Chen, "Weakly supervised metric learning towards signer adaptation for sign language recognition," in *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, September 2015, pp. 35.1–35.12. [Cited on page 104.]

[171] J. Zieren and K.-F. Kraiss, "Robust person-independent visual sign language recognition," in *Pattern Recognition and Image Analysis*, J. S. Marques, N. Pérez de la Blanca, and P. Pina, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 520–528. [Cited on page 104.]

[172] T. Shanableh and K. Assaleh, "User-independent recognition of arabic sign language for facilitating communication with the deaf community," *Digital Signal Processing*, vol. 21, no. 4, pp. 535 – 542, 2011. [Cited on page 104.]

[173] U. von Agris, J. Zieren, U. Canzler, B. Bauer, and K.-F. Kraiss, "Recent developments in visual sign language recognition," *Universal Access in the Information Society*, vol. 6, no. 4, pp. 323–362, Feb 2008. [Cited on page 104.]

[174] W. Kong and S. Ranganath, "Towards subject independent continuous sign language recognition: A segment and merge approach," *Pattern Recognition*, vol. 47, no. 3, pp. 1294 – 1308, 2014, handwriting Recognition and other PR Applications. [Cited on pages 104 and 105.]

[175] D. Kelly, J. McDonald, and C. Markham, "A person independent system for recognition of hand postures used in sign language," *Pattern Recognition Letters*, vol. 31, no. 11, pp. 1359 – 1368, 2010. [Cited on pages 104, 105, 112, and 113.]

[176] D. Dahmani and S. Larabi, "User-independent system for sign language finger spelling recognition," *Journal of Visual Communication and Image Representation*, vol. 25, no. 5, pp. 1240 – 1250, 2014. [Cited on pages 104, 112, and 113.]

[177] F. Yin, X. Chai, and X. Chen, "Iterative reference driven metric learning for signer independent isolated sign language recognition," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 434–450. [Cited on pages 104 and 105.]

[178] J. Triesch and C. von der Malsburg, "A system for person-independent hand posture recognition against complex backgrounds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 12, pp. 1449–1453, Dec. 2001. [Cited on page 110.]

[179] A. Just, Y. Rodriguez, and S. Marcel, "Hand posture classification and recognition using the modified census transform," in *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, April 2006, pp. 351–356. [Cited on pages 110, 112, and 113.]

[180] P. M. Ferreira, J. S. Cardoso, and A. Rebelo, "On the role of multimodal learning in the recognition of sign language," *Multimedia Tools and Applications*, Sep 2018. [Cited on pages 111, 112, 113, and 152.]

[181] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *CoRR*, vol. abs/1503.03832, 2015. [Cited on pages 111 and 112.]

[182] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008. [Cited on pages xix, 114, 154, and 155.]

[183] "Chaos Computer Clubs breaks iris recognition system of the Samsung Galaxy S8," www.ccc.de/en/updates/2017/iriden, accessed on 29th of May of 2017. [Cited on page 115.]

[184] R. Raghavendra and C. Busch, "Robust scheme for iris pres. attack det. using multiscale binarized statistical image features," *IEEE TIFS*, vol. 10, no. 4, pp. 703–715, 2015. [Cited on pages 115, 117, and 119.]

[185] A. Czajka and K. W. Bowyer, "Pad for iris recognition: An assessment of the state-of-the-art," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 86:1–86:35, Jul. 2018. [Cited on page 115.]

[186] J. Galbally, J. Fierrez, and R. Cappelli, *An Introduction to Fingerprint Presentation Attack Detection*. Cham: Springer International Publishing, 2019, pp. 3–31. [Cited on page 115.]

[187] U. Scherhag, C. Rathgeb, J. Merkle, R. Breithaupt, and C. Busch, "Face recognition systems under morphing attacks: A survey," *IEEE Access*, vol. 7, pp. 23 012–23 026, 2019. [Cited on page 115.]

[188] "Facetec liveness detection technology is ibeta/NIST certified anti-spoofing – level 1&2," www.zoomlogin.com/, accessed on 10th of June of 2019. [Cited on page 116.]

[189] A. F. Sequeira, S. Thavalengal, J. Ferryman, P. Corcoran, and J. S. Cardoso, "A realistic evaluation of iris presentation attack detection," in *39th TSP*, June 2016, pp. 660–664. [Cited on pages 116, 117, and 119.]

[190] E. Marasco and C. Sansone, "On the robustness of fingerprint liveness detect. alg. against new materials used for spoofing." in *BIOSIGNALS*, 2011, pp. 553–558. [Cited on page 117.]

[191] K. Bowyer and J. Doyle, "Cosmetic contact lenses and iris recognition spoofing," *Computer*, vol. 47, no. 5, pp. 96–98, May 2014. [Cited on page 117.]

[192] A. Rattani, W. Scheirer, and A. Ross, "Open set fingerprint spoof detection across novel fabrication materials," *IEEE TIFS*, vol. 10, no. 11, pp. 2447–2460, Nov 2015. [Cited on page 117.]

[193] A. F. Sequeira and J. S. Cardoso, "Fingerprint liveness detection in the presence of capable intruders," *Sensors*, vol. 15, pp. 14 615–14 638, 2015. [Cited on page 117.]

[194] A. F. Sequeira, J. Murari, and J. S. Cardoso, "Iris liveness detection methods in mobile applications," in *Proc. Int. Con. on CV Theory and Applic.*, 2014, pp. 22 – 33. [Cited on page 117.]

[195] S. R. Arashloo, J. Kittler, and W. Christmas, "An anomaly detection approach to face spoofing detection: A new formulation and evaluation protocol," *IEEE Access*, vol. 5, pp. 13 868–13 882, 2017. [Cited on page 117.]

[196] D. Menotti, G. Chiachia, A. Pinto, W. Robson Schwartz, H. Pedrini, A. Xavier Falcao, and A. Rocha, "Deeprep.iris,face,and fingerp.spoof.det." *TIFS*, vol. 10, no. 4, pp. 864–879, 2015. [Cited on page 117.]

[197] A. Pinto, H. Pedrini, M. Krumdick, B. Becker, A. Czajka, K. W. Bowyer, and A. Rocha, "Counteracting presentation attacks in face, fingerprint, and iris recognition," *Deep Learning in Biometrics*, vol. 245, 2018. [Cited on page 117.]

[198] R. Tolosana, M. Gomez-Barrero, J. Kolberg, A. Morales, C. Busch, and J. Ortega-Garcia, "Towards fingerprint presentation attack detection based on convolutional neural networks and short wave infrared imaging," in *2018 International Conference of the Biometrics Special Interest Group (BIOSIG)*. IEEE, 2018, pp. 1–5. [Cited on page 117.]

[199] H. Zhang, Z. Sun, and T. Tan, "Contact lens detection based on weighted LBP," in *20th ICPR*, 23 - 26 August 2010, pp. 4279–4282. [Cited on page 119.]

[200] J. Hu, J. Lu, Y. Tan, and J. Zhou, "Deep transfer metric learning," *IEEE Transactions on Image Processing*, vol. 25, no. 12, pp. 5576–5588, Dec 2016. [Cited on pages 126, 127, 128, 155, and 157.]

[201] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Józefowicz, and S. Bengio, "Generating sentences from a continuous space," *CoRR*, vol. abs/1511.06349, 2015. [Cited on page 151.]

[202] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987. [Cited on page 155.]

[203] J. C. Dunn, "A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters," *Journal of Cybernetics*, vol. 3, no. 3, pp. 32–57, 1973. [Cited on page 155.]