Transfer Learning methods for temporal data

Guillaume Richard

université
PARIS-SACLAY

Méthodes d'apprentissage statistique de type "Transfer Learning" pour des données temporelles multivariées

Transfer Learning for Temporal Data

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n° 574, Ecole Doctorale de Mathématiques Hadamard (EDMH)

Spécialité de doctorat: Mathématiques Appliquées

Unité de recherche: Centre Borelli

Référent: ENS Paris-Saclay

Thèse présentée et soutenue à Gif-sur-Yvette,

le 7 octobre 2021, par

Guillaume RICHARD

Composition du jury

Younès Bennani Professeur, Université Sorbonne Paris Nord (Laboratoire d'Informatique de Paris Nord	Président et Rapporteur
Gianluca Bontempi Professeur, Université Libre de Bruxelles (Département d'Informatique)	Rapporteur
Jairo Cugliari Maitre de Conféerences, Université Lumière 2 (Laboratoire ERIC)	Examinateur
Michèle Sebah Directrice de Recherche, INRIA (Equipe TAO)	Examinatrice

Direction de la thèse

Mathilde Mougeot Professeur, ENS Paris-Saclay (Centre Borelli) et ENSIIE	Directrice
Nicolas Vayatis Professeur, ENS Paris-Saclay (Centre Borelli)	Directeur
Georges Hébrail Chercheur Senior, EDF R&D	Tuteur en entreprise

Thèse de doctorat

NNT: 2021UPASM037

Remerciements

Mes premiers remerciements vont à mes directeurs de thèse, Mathilde Mougeot et Nicolas Vayatis pour leur confiance au long de ces trois années de thèse, et de m'avoir donné l'opportunité de travailler sur des sujets aussi intéressants. Je remercie également Georges Hébrail qui a su m'accompagner dès mes premiers pas chez EDF et dépasser son rôle d'encadrant industriel pour toujours me donner des bons conseils. Je remercie également tous les membres du jury pour avoir accepté de participer à ma soutenance de thèse. En particulier, je remercie Younès Bennani qui a accepté de présider le jury en plus d'avoir rapporté ma thèse, Gianluca Bontempi qui a lui aussi rapporté ma thèse et enfin Jairo Cugliari et Michèle Sebag en tant qu'examinateurs.

Ensuite, je souhaite remercier toute l'équipe SOAD d'EDF pour tous les bons moments passés ensemble. J'aurai certainement l'occasion de le dire de vive voix, mais c'était un réel plaisir de travailler avec vous. En plus de Georges, j'aimerais remercier Ghislain qui m'a accompagné sur la fin de la thèse. Quoiqu'il en soit, j'espère qu'on sera amené à retravailler ensemble et que je serai toujours invité au Diamant!

Je veux aussi remercier l'ensemble des membres du Centre Borelli, en particulier les thésards. L'entraide entre les doctorants et nos discussions ont permis d'égayer les moments difficiles. En particulier, je remercie Antoine avec qui c'était un plaisir de collaborer et je te souhaite le meilleur pour le reste de ta thèse.

Enfin, je veux remercier mes amis et ma famille, qui m'ont soutenu tout au long de la thèse et m'ont aidé à me changer les idées. En particulier, je remercie ma mère, mon père et mon petit frère qui m'ont fait grandir, m'ont déjà accompagné dans toutes les étapes importantes de ma vie et continueront à le faire. Finally, I want to thank you, Quỳnh Anh, for your love and support. It really made it easier to overcome every difficult moment and to move to the next chapter together.

Contents

List of Figures

List of Tables

Résumé (en français)

Contexte de la thèse

Cette thèse de doctorat propose de développer des méthodes d'apprentissage par transfert (*transfer learning*) pour des séries temporelles. Ce travail a été réalisé dans le cadre du dispositif CIFRE (Convention Individuelle de Formation par la Recherche) de l'Agence Nationale de la Recherche et de la Technologie (ANRT), a été sponsorisé par Electricité De France (EDF), premier producteur et fournisseur d'électricité en Europe, et co-encadré par le Centre Borelli de l'Ecole Normale Supérieure Paris-Saclay. EDF emploie plus de 160 000 salariés en 2020 et fournit la majorité des clients résidentiels et industriels en France. En tant que producteur et fournisseur d'électricité, EDF se doit de toujours améliorer ses capacités de production ainsi que sa connaissance de la consommation de ses clients. Ansi, l'analyse de signaux temporels est au coeur de nombreuses problématiques rencontrées par les ingénieurs d'EDF, par exemple pour la surveillance de machines en production ou la prédiction de consommation des clients. Dans ce cadre, les algorithmes d'apprentissage automatique ont prouvé qu'ils pouvaint répondre à ces problématiques mais nécessitent bien souvent de larges bases de données d'apprentissage, souvent coûteuses à collecter. L'apprentissage par transfert englobe plusieurs méthodes visant à capitaliser sur un modèle ou une base de données déjà existants et pourrait permettre de réduire le besoin de collecte de nouvelles données. Dans ce contexte, ce travail propose d'étudier et de proposer de nouvelles méthodes d'apprentissage par transfert pour des données temporelles, avec en visée des applications sur l'étude de la consommation d'un foyer et la surveillance de machines par des capteurs en production.

Motivations scientifiques et industrielles

Autour de nous, de nombreuses activités impliquent une variation dans le temps : les prix du pétrole changent toutes les milli-secondes, la consommation d'énergie varie toutes les secondes, la propagation d'un virus est aussi un phénomène temporel ... L'analyse de ces phénomènes qui dépendent du temps est le champ d'étude de l'analyse des séries chronologiques. Chez EDF, les phénomènes variant dans le temps apparaissent dans de nombreux scénarios. En tant que producteur d'électricité, l'un des principaux défis d'EDF est d'assurer l'adéquation entre la

production et la consommation d'électricité, tant pour des raisons de sécurité que pour des raisons économiques. Alors qu'EDF contrôle la production d'électricité et peut l'ajuster, la consommation dépend des clients, qui sont externes à l'enterpise et donc non contrôlés.

Une meilleure connaissance des habitudes de consommation de ses clients aurait plusieurs autres avantages pour EDF. Le marché français de l'électricité ayant été ouvert à la concurrence en 2010, EDF souhaite proposer des contrats de plus en plus adaptés à ses clients. De plus, l'autoproduction d'électricité via une éolienne ou des panneaux solaires attachés à un foyer prend de plus en plus d'importance. Il est donc essentiel de comprendre la consommation au niveau des ménages pour mieux organiser le réseau électrique, car les données de consommation d'un ménage sont traditionnellement suivies à l'aide d'un seul compteur.

Les compteurs historiques ne peuvent pas être lus à distance et ne sont relevés qu'une fois par an par un technicien. Depuis 2015, les compteurs communicants Linky sont déployés en France pour remplacer les anciens compteurs. Ces compteurs collectent la consommation d'électricité toutes les 30 minutes et les données de consommation sont envoyées à EDF avec l'accord du client. Cependant, même avec les compteurs Linky, l'accès aux données de consommation des ménages reste limité, en raison des coûts de traitement et de stockage ansi que de problèmes de confidentialité. De plus, les données à pas 30 minutes donnent un aperçu général de la consommation d'un ménage mais ne fournissent pas de détails sur l'utilisation des différents appareils.

La désagrégation de la courbe de charge (consommation d'électricité) d'un ménage en fonction de l'utilisation de chaque appareil est le sujet de la surveillance non intrusive de la charge (NILM), illustrée dans la figure 1.1. Le NILM peut donner un aperçu complet de la consommation d'un client et pourrait permettre à EDF de proposer des contrats parfaitement adaptés aux besoins d'un client. Mais Le NILM nécessite des données échantillonnées de 1 Hz à plusieurs kHz, un échantillonage bien supérieur à celui de 30 minutes de Linky.

Par conséquent, la collecte de données pour Le NILM est très coûteuse : non seulement la consommation de l'ensemble du ménage doit être mesurée toutes les secondes, mais aussi la consommation de chaque appareil présent dans le foyer. Cela signifie que pour collecter des données pour une seule maison, un technicien doit venir à la maison pour installer les capteurs sur chaque appareil. Il serait irréaliste de le faire à grande échelle et aucun ensemble de données publiques collectées pour le NILM ne contient plus d'une centaine de ménages. Nous devons donc concevoir des méthodes de le NILM qui soient robustes à un changement de foyer.

Les séries temporelles apparaissent aussi dans les systèmes de production d'EDF car chaque système industriel est surveillé par des capteurs. Dans les centrales électriques, plusieurs machines sont surveillées pour détecter des comportements anormaux et planifier une maintenance. Pour les éoliennes ou les panneaux solaires, la prévision de la production est le principal défi. Lors de l'apprentissage d'un modèle de prévision sur un système donné, le transfert vers un autre système peut entra1ner une baisse de performance dûe potentiellement à un changement d'environnement ou un changement de fonctionnement après une maintenance.

Dans cette thèse, nous abordons la question de l'apprentissage par transfert pour les séries temporelles.

L'objectif est de concevoir des méthodes capables d'extraire des caractéristiques de séries temporelles robustes à un changement d'environnement. Bien que les méthodes développées dans cette thèse soient développées pour répondre à des problématiques d'EDF, notre cadre est générique à d'autres types de problématiques.

L'apprentissage par transfert [138] est un sous-domaine de l'apprentissage automatique ayant reçu beaucoup d'attention au cours de la dernière décennie. Une des principales hypothèses de l'apprentissage automatique traditionnel est que les données utilisées pour apprendre un modèle suivent la même distribution que les données auxquelles le modèle sera appliqué. L'apprentissage par transfert suppose que cette hypothèse n'est pas vérifiée et propose de surmonter ce problème. L'apprentissage d'un nouveau modèle sur le nouvel environnement n'est pas toujours possible, par exemple si la collecte de données est coûteuse. L'apprentissage par transfert vise à capitaliser sur les données ou les modèles existants pour s'adapter plus rapidement à un nouvel environnement. Une analogie avec l'apprentissage humain serait qu'un guitariste confirmé apprendra plus rapidement à jouer du piano qu'un débutant complet.

Une série temporelle est un ensemble de mesures qui varient dans le temps. Elles sont particulièrement difficiles à réaliser car elles sont souvent de grande dimension, soumises au bruit et la dépendance temporelle brise l'hypothèse d'indépendance faite par de nombreux modèles d'apprentissage statistique. C'est pourquoi des méthodes spécifiques aux séries temporelles ont été développées. En général, les séries temporelles brutes sont transformées en de nouvelles représentations, soit par une transformation experte telle que la transformée de Fourier [31] ou la transformée en ondelettes [119], soit par l'apprentissage de cette transformation.

Dans cette thèse, nous proposons de coupler apprentissage par transfert et analyse des séries temporelles.

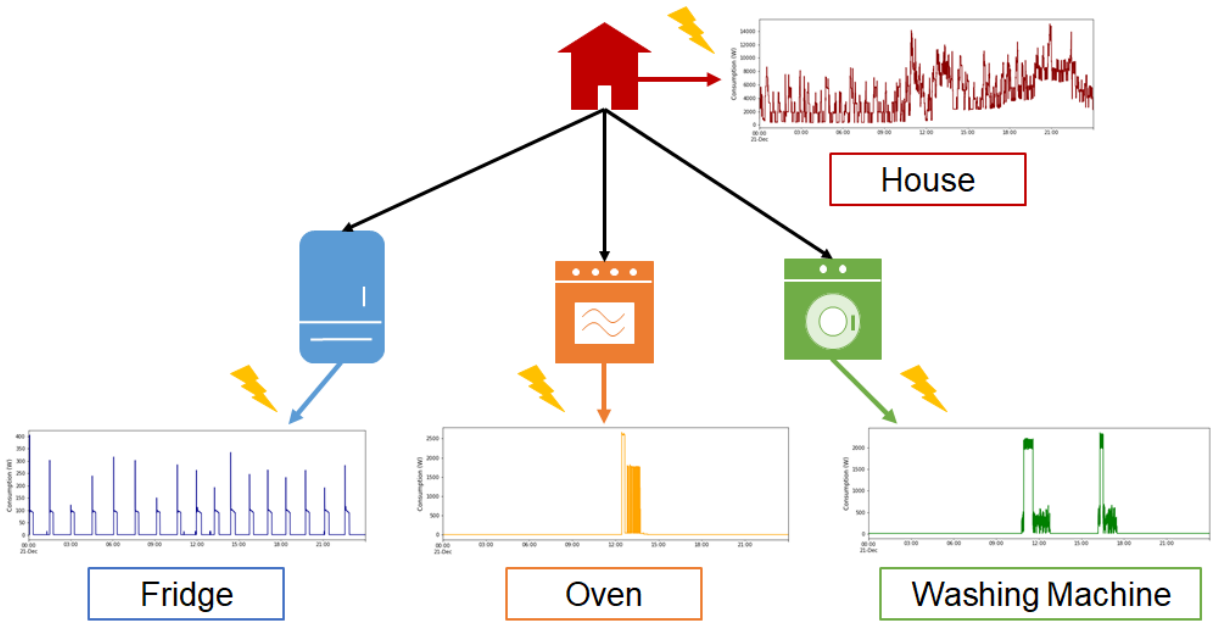


Figure 1: Illustration du NILM: à partir de la consommation totale de la maison, l'objectif est de retrouver la consommation de chaque appareil.

En effet, si ces deux sujets ont suscité beaucoup d'intérêt dans la communauté de l'apprentissage automatique, le développement de méthodes d'apprentissage par transfert pour les séries temporelles n'a suscité qu'un nombre limité de travaux [64]. L'objectif principal de cette thèse peut être formulé comme suit :

Pouvons-nous développer des méthodes d'extraction de features transférables à partir de séries temporelles ?

Les méthodes présentées dans cette thèse s'appuient sur l'analyse traditionnelle des séries chronologiques et transfèrent l'apprentissage afin de réduire la nécessité de collecter de nouvelles données après un changement d'environnement.

Contributions

Dans cette thèse, nous nous intéressons au problème de l'apprentissage par transfert appliqué aux séries chronologiques. Nous étudions notamment le cadre dans lequel un ensemble de données "source" est disponible, mais nous voulons apprendre un modèle capable de généraliser à un ensemble de données "cible" Par exemple, le jeu de données source sera un ménage où la consommation de chaque appareil est mesurée et la cible est une nouvelle maison, avec différents appareils et potentiellement une différente composition du foyer. Ce changement est représenté par une modification de la distribution des données. Il est clair que si la composition des maisons est radicalement différente, l'algorithme appris sur la maison source a peu de chances de fonctionner correctement sur la maison cible.

Afin de pouvoir généraliser à une nouvelle maison, différentes approches sont possibles. La généralisation est un problème de longue date dans l'apprentissage automatique, et plusieurs travaux ont proposé des garanties sur la capacité de généralisation des méthodes d'apprentissage statistique lorsqu'il n'y a pas de changement dans la distribution [131] [176]. Lorsqu'il y a un changement d'environnement, ces garanties ne sont plus valables. C'est pourquoi de nouveaux travaux proposent des garanties d'apprentissage avec un changement de distribution. L'un des thèmes abordés dans le cadre d'un changement d'environnement est l'apprentissage de l'invariance ou des causalités [142]. En général, l'étude des causalités formule le problème de l'apprentissage des relations entre différentes variables qui sont robustes au changement d'environnement.

L'apprentissage par transfert couvre un large éventail d'hypothèses et de méthodes, allant de la recherche d'une invariance entre plusieurs environnements à l'adaptation à un nouvel environnement. Dans cette thèse, nous nous intéressons au cadre d'adaptation de domaine, qui diffère légèrement de l'apprentissage d'invariance. L'adaptation de domaine suppose que la nature des données est la même pour le domaine source et le domaine cible, *ie* les variables d'entrée mesurées et la variable à prédire sont de même nature dans chaque domaine. De plus, l'objectif de l'adaptation de domaine est d'extraire du domaine source les connaissances qui sont pertinentes pour le domaine cible. Cela doit être fait avec seulement des données non étiquetées ou un petit nombre de données

étiquetées dans le jeu de données cible.

Différents travaux ont présenté des garanties pour l'adaptation de domaine [24] [121]. Dans le chapitre 2, nous présentons les principales bornes de la littérature pour l'adaptation de domaine. Nous passons également en revue plusieurs méthodes d'adaptation de domaine afin de donner un aperçu de la manière de réduire le décalage entre les différents ensembles de données. Dans ce chapitre, nous présenterons également l'adaptation à partir de plusieurs sources. En effet, lors de l'adaptation à partir de plusieurs domaines, il est possible d'extraire davantage d'information. Une première possibilité est d'utiliser les différents domaines pour extraire une invariance entre les domainesafin de généraliser à un nouveau domaine de même nature. On peut également essayer d'apprendre quels sont les domaines sources les plus pertinents pour la cible. Par exemple, si plusieurs maisons sont surveillées et utilisées comme domaines sources, lors de l'adaptation à une nouvelle maison cible, on s'attend à sélectionner les maisons sources qui ont des caractéristiques similaires à la cible.

La plupart des méthodes traditionnelles d'adaptation de domaine reposent sur la réduction d'un décalage entre les distributions source et cible. Lorsque l'on travaille avec des vecteurs $x \in \mathbb{R}^d$ de dimension d et un échantillon indépendamment et identiquement distribué (iid) $\{x^{(1)}, ..., x^{(n)}\}$, il est possible d'estimer la distribution de l'échantillon avec des estimateurs courants tels que l'estimateur de densité de noyau. Lorsque l'on travaille avec une série temporlle de dimensions d $\{x_1, ..., x_T\}$ où $x_t \in \mathbb{R}^d$, l'estimation de sa distribution n'est pas simple. On pourrait considérer x comme un vecteur de dimension dT mais très souvent, T est plus grand que la taille de l'échantillon n. En effet, les séries temporelles brutes sont souvent de haute dimension, soumises au bruit et ont une structure spécifique avec leur dépendance temporelle.

Pour ces raisons, il est courant de transformer les séries temporelles brutes en un nouvel espace de représentation. Dans le Chapitre 3, nous passons en revue certaines méthodes d'extraction de caractéristiques courantes pour les séries chronologiques. En particulier, nous nous intéressons aux réseaux de neurones qui ont suscité beaucoup d'intérêt avec leur succès dans la communauté de la vision par ordinateur [103]. Il a été démontré que les réseaux de neurones sont capables d'extraire des caractéristiques générales pour des données d'images ou de texte lorsqu'elles sont apprises sur de grands ensembles de données. Nous verrons que pour les séries temporelles, il est plus difficile d'évaluer ces capacités de généralisation car les séries temporelles sont présentent dans de nombreux domaines très hétérogènes et aucun grand ensemble de données étiquetées n'est disponible à ce jour.

Dans ce chapitre, nous étudions le problème de la reconnaissance des appareils, *ie* reconna1̂tre un appareil à partir d'un de ses cycles de consommation (aussi appelé signature de l'appareil). Nous discutons de différentes techniques de normalisation pour les séries temporelles et montrons que pour trouver des invariants entre les ensembles de données, des méthodes spécifiques sont nécessaires. Nous introduisons une méthode d'ensemble de normalisation qui donne une meilleure robustesse à changement de domaine. Nous illustrons nos résultats sur des jeux de données de NILM collectés à la fois en interne chez EDF et des jeux de données publics.

Encouragés par la capacité des réseaux de neurones à apprendre des représentations informatives pour les séries temporelles, nous proposons une méthode d'adaptation de domaine basée sur les réseaux de neurones dans le Chapitre 4. Notre méthode s'inspire de l'adaptation de domain adverse (*domain-adversarial adaptation*) [72] : l'objectif formulé vise à apprendre une représentation qui (i) soit discriminante pour la variable à prédire et (ii) réduise l'écart entre les domaines. Alors que la plupart des travaux précédents se concentrent sur des problèmes de classification, le NILM a pour but de trouver la consommation (une grandeur réelle) et est généralement un problème de régression [100]. Nous proposons une nouvelle mesure de divergence entre les domaines, une nouvelle borne théorique associée et un algorithme qui propose une vision unifiée de nombreuses méthodes d'adaptation de domaine adverse antérieures. Cette garantie est étendue à l'adaptation multi-source, pour laquelle nous proposons d'attribuer des poids aux sources en fonction de leur relation avec le domaine cible. Nous illustrons nos résultats sur plusieurs jeux de données de NILM, mais aussi sur d'autres problèmes typiques d'adaptation au domaine : (i) analyse de sentiment des commentaires textuels sur Amazon [29] et (ii) labellisation d'images de chiffres (MNIST, MNIST-M, ...) [72].

Si les méthodes basées sur les réseaux de neurones donnent de très bons résultats expérimentaux, elles souffrent de deux problèmes : elles nécessitent généralement beaucoup de données d'entraînement et sont difficilement interprétables. En effet, il est difficile de visualiser les représentations apprises par le réseau et de les relier à des concepts interprétables. Il est donc difficile de les intégrer dans des applications industrielles car très souvent, les opérateurs veulent comprendre le comportement d'un modèle. C'est pourquoi, dans le chapitre 5, nous proposons une méthode d'apprentissage par transfert pour les séries temporelles multivariées basée sur les informations de covariance.

Plus précisément, étant donné que plusieurs systèmes sont surveillés par des capteurs d, nous proposons de construire un sous-espace virtuel de capteurs de dimension $d' < d$ de telle sorte que les relations entre les dimensions extraites soient les mêmes dans chaque système. Ainsi, nous proposons de séparer les caractéristiques communes à tous les systèmes des caractéristiques spécifiques à chaque système. Notre cadre d'étude intègre donc à la fois la généralisation de domaine et à l'adaptation à un domaine précis. Comme nous ne travaillons que sur les matrices de covariance provenant de séries temporelles multivariées, le nouvel espace de représentation se base sur la géométrie spécifique des matrices symétriques à définition positive. Malheureusement, durant ce travail de thèse, aucun jeu de données industrielles à EDF ne contenait suffisamment de données étiquetées pour expérimenter notre méthode. Par conséquent, nous expérimentons notre méthode sur des données synthétiques et des jeux de données de reconnaissance d'activité humaine afin de montrer les performances et les limites de notre approche.

Ces différentes contributions ont donné lieu à des publications dans des conférences nationales et internationales :

- Chapitre 3 G. Richard, G. Hébrail, M. Mougeot and N. Vayatis. ”DenseNets for Time Series Classification: towards automation of time series pre-processing with CNNs.” MiLeTS19@SIGKDD 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD’2019).
- Chapitre 3 G. Richard, B. Grossin, G. Germaine, G. Hébrail and A. de Moliner. ”Autoencoder-based time series clustering with energy applications.” Conférence sur l’Apprentissage Automatique 2018 (CAp 2018).
- Chapitre 4 G. Richard, A. de Mathelin, G. Hébrail, M. Mougeot and N. Vayatis. ”Unsupervised Multi-Source Domain Adaptation for Regression.” Joint European Conference on Machine Learning and Knowledge Discovery in Databases 2020 (ECML 2020)
- Chapitre 4 A. de Mathelin, G. Richard, M. Mougeot, and N. Vayatis. ”Adversarial Weighting for Domain Adaptation in Regression.” arXiv preprint arXiv:2006.08251. (Pre-print)

Enfin, les expériences menées sur les ensembles de données publiques pour cette thèse sont accessibles, sous la forme d’une contribution au paquet *adapt* et d’un dépot git. Nous donnons dans l’Annexe C des détails sur les différentes implémentations pour la reproduction des résultats obtenus sur des jeux de données publiques.

Chapter 1

Introduction

1.1 Motivation

As an electricity producer, one of the main challenges of EDF is to ensure the matching between electricity production and electricity consumption, both for safety and economical reasons. While EDF controls the production of electricity and can adjust it, the consumption depends on external clients. Hence, EDF aims to enhance its knowledge of electricity consumption of its users.

A better understanding of the consumption patterns of its clients would have several other upsides for EDF. As the French electricity market has been open to competitors in 2010, EDF wants to offer contracts tailored for its clients. Moreover, self-production of electricity via a wind plant or solar panels attached to a household is larger everyday. Hence, understanding consumption at a household level is crucial to a better organization of the electricity network.

Consumption data of a household is traditionally monitored using a single sensor. The historical sensor meters could not be read remotely and were only read once per year by an EDF employee. Since 2015, Linky smart sensors are being deployed in France to replace the previous meters. These sensors collect the electricity consumption every 30 minutes and the consumption data is sent to EDF with the client's consent. Still, even with Linky sensors, access to household consumption data is still limited, due to transmission and processing costs or privacy issues.

Moreover, 30 minute data gives a general overview of a household consumption but does not provide details about the usage of different devices. Disagreggating the load curve (electricity consumption) of a household to the usage of each device is the topic of Non Intrusive Load Monitoring (NILM), illustrated in Figure 1.1. NILM may give a complete overview of the consumption of a client and could allow EDF to propose contracts perfectly tailored to clients' needs. But NILM requires data that is sampled from 1Hz to several kHz, which is far from the 30 minute sampling of Linky.

Hence, collecting training data for NILM is even more costly: not only do we need to monitor the whole household

consumption at a second interval, but we also need to monitor each device in the household. This means that for collecting data for only one house, a technician needs to come to the home to install the sensors on every device which comes at an important cost. It would be unrealistic to do it on a large scale and no public dataset collected for NILM contained more than a hundred households. Hence, we need to design NILM methods that are robust to a change of household.

Time series can also arise in production systems at EDF as every industrial system is monitored with sensors. In power plants, several machines are monitored to detect abnormal behaviours and plan maintenance. In wind plants or solar panels, production forecasting is the main challenge. When learning a prediction model on a given system, transferring it to a different system can lead to a drop in performance given to a distribution shift. It could also happen on the same system after a maintenance.

In this thesis, we tackle the issue of statistical learning on time series with a distribution shift, also called transfer learning for time series. The objective is to design methods able to extract features robust to a change of environment. While the methods developed in this thesis are applied to EDF applications, our framework is generic to other kind of temporal data.

Transfer Learning [138] is a sub-field of Machine Learning that received a lot of attention in the last decade. In traditional Machine Learning, one of the main assumption is that the data used to train a model follows the same distribution as the data to which the model will be applied. In Transfer Learning, this assumption does not hold. Learning a new model on the new environment is not always possible, either for instance if data collection is expensive. Transfer Learning aims to capitalize on existing data or models to adapt more quickly to a new

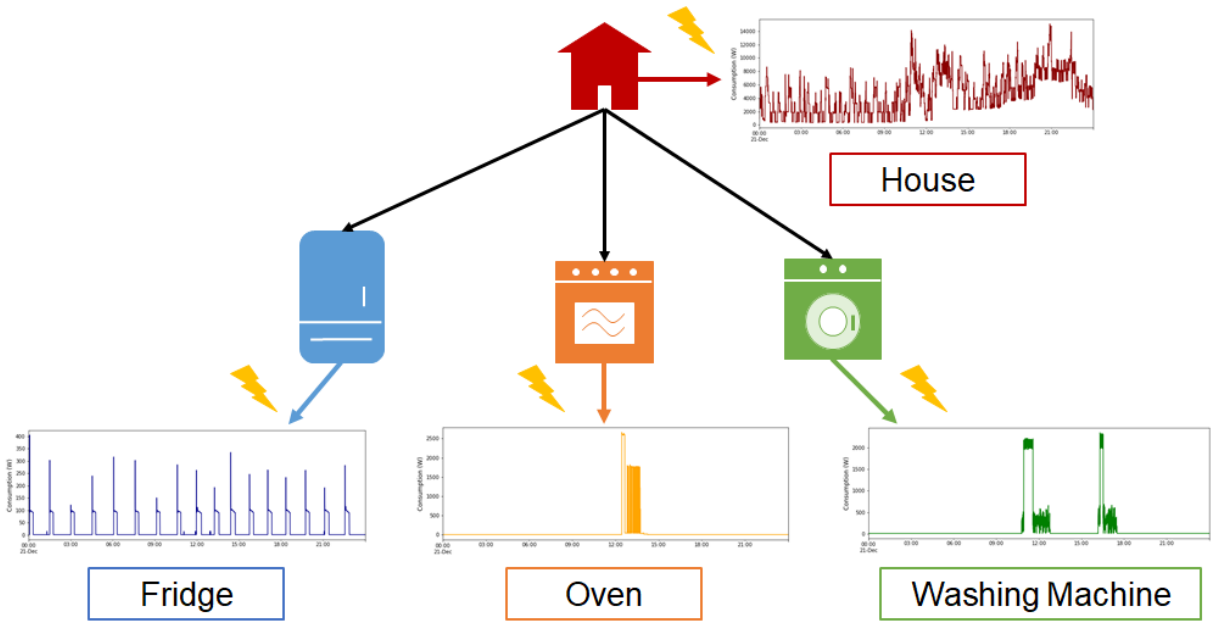


Figure 1.1: Non Intrusive Load Monitoring illustration: from the whole house consumption, the goal is to retrieve the consumption of each appliance

environment. An analogy with human learning would be that a proved guitarist will learn more quickly how to play the piano than a complete beginner.

Time Series analysis is a long-standing issue in the data mining community, with applications in many fields. A time series is a collection of measurements that vary over time. They are particularly challenging as they often are high-dimensional, subject to noise and the temporal dependence breaks the independence assumption made by many statistical learning models. Hence, methods specific to time series were developed. In general, the raw time series are transformed to new features, either by a hand-crafted transformation such as Fourier Transform [31] or Wavelet Transform [119] or by learning this transformation.

In this thesis, we match Transfer Learning and Time Series analysis. Indeed, while both topics attracted a lot of interest in the machine learning community, developing transfer learning methods for time series has prompted only a limited amount of works [64]. The main goal of this thesis can be formulated as follows:

Can we develop dedicated methods to extract transferable features from time series?

1.2 Organization of the manuscript

Transfer Learning covers a wide range of frameworks and methods, from finding invariance to adapting to a new environment. In this thesis, we are interested in the domain adaptation framework, which is slightly different from learning invariance. Domain adaptation assumes that the nature of the data is the same for the source and the target domain, *ie* the measured input variables and the variable to predict are of the same nature in every domain. Moreover, in domain adaptation, the goal is to extract knowledge from the source domain that is relevant to the target domain. This needs to be done with only unlabeled data or a few labeled data in the target dataset.

In Chapter 2, we present the main domain adaptation guarantees of the literature [24] [121]. We also review several domain adaptation methods to give an overview of how to reduce the shift between different datasets. In that Chapter, we will also introduce adaptation from multiple sources as specific methods can be developed for this scenario. Firstly, one could use the several domains to learn invariance between domains, which is called domain generalization. One could also try to learn which source domains are the most relevant to the target. For instance, if several houses are monitored and used as a source domains, when adapting to a new target house, one would expect to select the source houses that have similar characteristics to the target. This is called multi-source domain adaptation.

Most traditional methods of domain adaptation relies on reducing a shift between the source and target distributions. When working with d-dimensional vectors $x \in \mathbb{R}^d$ and an independently identically distributed (iid) sample $\{x^{(1)}, ..., x^{(n)}\}$, estimating the distribution of the sample is possible with common estimators such as Kernel Density Estimator. When working with a d-dimensional time series $\{x_1, ..., x_T\}$ where $x_t \in \mathbb{R}^d$, estimating its distribution is

not straightforward. One could vectorize x to a vector of dimension dT but very often, T is larger than the sample size n. Indeed, raw time series are often high-dimensional, subject to noise and have a specific structure with their temporal dependence.

For these reasons, it is common to transform raw time series to a new feature space. In Chapter 3, we will review some common feature extraction methods for time series. In particular, we are interested in neural networks which gained a lot of interest with their success in the computer vision community [103]. Neural networks have been shown to be able to extract general features for image or text data when learnt on large datasets. We will see that for time series, it is harder to evaluate those generalization capabilities as time series are heterogeneous as they arise in many fields and no large labeled dataset is available.

We will study the problem of appliance recognition, *ie* recognize an appliance from its consumption patterns. We discuss different normalization techniques for time series and show that in order to find invariants between datasets, specific methods are required. We introduce a normalization ensembling method that allows for better robustness when changing domain. We illustrate our results on non intrusive load monitoring datasets both collected at EDF and publicly available.

Encouraged by the ability of neural networks to learn informative features for time series, we propose a domain adaptation method based on neural networks in Chapter 4. It follows the line of adversarial domain adaptation [72]: the formulated objective aims to learn features that are (i) discriminative for the variable to predict and (ii) reducing the shift between domains. While most previous works focus on classification problems, Non-Intrusive Load Monitoring aims to find the consumption (a real value) and is generally a regression problem [100]. We propose a new measure of discrepancy between domains, a new theoretical bound and an algorithm that offers a unifying view over many previous adversarial domain adaptation methods. We extend this guarantee to adaptation with multiple sources, where we propose to attribute weights to the sources depending on their relationship with the target domain. We illustrate our results on several NILM datasets, but also on other standard domain adaptation datasets: Amazon Multi-Domain Sentiment dataset [29] (text) and digits dataset [72] (images).

While neural network based methods give very good experimental results, they suffer from two issues: they generally require a lot of training data and are hardly interpretable. It makes it hard to integrate them in industrial applications, as very often, operators want to understand the behaviour of a model. Hence, in Chapter 5, we propose a transfer learning method for multivariate time series based on covariance information.

Namely, given several systems that are monitored with d sensors we propose to build a virtual sensor subspace of dimension $d' < d$ such the relationship between the extracted dimensions are the same in every system. As such, we propose to separate the knowledge that is transferable between every domain from the domain-specific knowledge. Our framework is applicable both to domain generalization and domain adaptation. As we only work on covariance information from multivariate time series, the new feature space lies on the manifold of Symmetric Positive Definite matrices. Our method makes use of the specific geometry of this manifold. Unfortunately, at the

time of this thesis, no industrial dataset at EDF contained enough labeled data to experiment our method. Hence, we experiment our method on synthetic data and Human Activity Recognition datasets to show the strengths and limits of our approaches.

Those different contributions led to 3 publications in peer-reviewed national and international conferences and a pre-print:

- Chapter 3 G. Richard, G. Hébrail, M. Mougeot and N. Vayatis. "DenseNets for Time Series Classification: towards automation of time series pre-processing with CNNs." MiLeTS19@SIGKDD 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'2019).
- Chapter 3 G. Richard, B. Grossin, G. Germaine, G. Hébrail and A. de Moliner. "Autoencoder-based time series clustering with energy applications." Conférence sur l'Apprentissage Automatique 2018 (CAp 2018).
- Chapter 4 G. Richard, A. de Mathelin, G. Hébrail, M. Mougeot and N. Vayatis. "Unsupervised Multi-Source Domain Adaptation for Regression." Joint European Conference on Machine Learning and Knowledge Discovery in Databases 2020 (ECML 2020)
- Chapter 4 A. de Mathelin, G. Richard, M. Mougeot, and N. Vayatis. "Adversarial Weighting for Domain Adaptation in Regression." arXiv preprint arXiv:2006.08251. (Pre-print)

The contribution of Chapter 5 will lead to a publication in the future. Finally, experiments conducted on public datasets for this thesis are publicly available, in the form of a contribution to the *adapt* package and public git repositories. We give in Appendix C details about the different implementations and how to reproduce the experiments on public datasets.

Chapter 2

Background on Transfer Learning

Contents

Classically, when learning a statistical model, one assumes that the data used for training have the same characteristics and distribution as the one on which the model is used. This assumption is often unrealistic. Consider a

fault detection model trained on an industrial system: when using it on a new industrial system operating in a different environment, the model under-performs. Transfer learning is a sub-field of Machine Learning where practitioners try to use knowledge learnt on a source problem to transfer it to a target problem. It has gained a lot of interest as often, data collection, annotation and storage for a new problem is expensive. Transfer learning has already proven its efficiency in the computer vision community: the knowledge learnt by an algorithm learnt to recognize a large number of objects can be used for more specific problems.

There are two natural questions arising with transfer learning, as was highlighted in [138]. The first question is on the guarantees of transfer learning: when can one expect a source problem to be relevant for a target domain? This is crucial as transferring knowledge from an unrelated source problem could lead to negative transfer, *ie* performing worse than learning directly on the target problem. The second natural question is on what and how to transfer this knowledge. The answer appears to be dependent on the nature of the problem.

In this chapter, we introduce the transfer learning framework and give an overview of the different transfer learning scenarios and recall their definition, as they are sometimes used interchangeably in the literature. As the main focus of this work is on domain adaptation, we review the existing theory of Domain Adaptation. Then we present an overview of domain adaptation methods categorized on what they use to adapt the learning model. Finally, we also define the concept of learning from multiple sources.

2.1 What is Transfer Learning?

Ideally, when predicting the consumption of a device in a house based on its total consumption, one would collect data from this house and apply the model on the same house. In a classical machine learning setting, practitioners are generally provided with a training sample in order to perform a prediction task. Then, a model learned to perform this task is assumed to have good generalization capacities on a test sample as it is assumed to come from the same distribution.

Transfer learning breaks one of the traditional machine learning assumptions: the domain or task to be learnt are different from the domain or task on which the model is applied. For instance, the input source data space might be different from the target one, the task could change or there could be a shift in the distributions. In the general transfer learning framework, one assumes the existence of a source domain and task and a target domain and task. If there is enough labeled data in the target domain, then one can simply use traditional machine learning to learn a model. But when there is not enough labeled target data, one wants to capitalize on the source data to extract knowledge that is transferable to the target data.

Following [138] and [185], we introduce a domain $\mathcal{D}$ made of a feature space $\mathcal{X}$ and a marginal probability distribution P, such that any independent sample $X = \{x_1, ..., x_n\}$ drawn from that domain is independently and identically distributed ($x_i \overset{i.i.d}{\sim} P$) and we write $\mathcal{D} = \{\mathcal{X}, P(X)\}$. We also introduce a task $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$ where

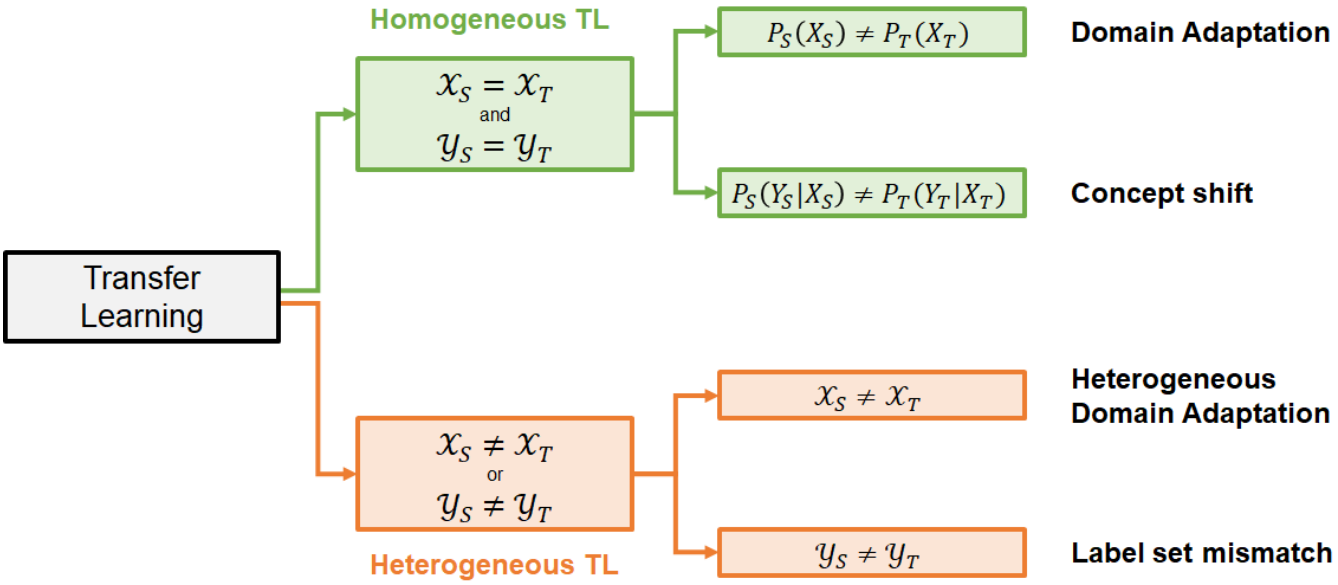


Figure 2.1: Taxonomy of Transfer Learning

$\mathcal{Y}$ is the label set (can be $\mathbb{R}$ for regression) and $P(Y|X)$ is the conditional probability of the label depending on the input data. Typically, one uses a sample $\{(x_1, y_1), ..., (x_n, y_n)\}$ where $x_i \sim P(X)$ and $y_i \sim P(Y|X = x_i)$ to learn a prediction function $\hat{f}$ minimizing the loss between $\hat{f}(x_i)$ and y_i where this loss is dependent on the problem. Then different works have shown that under certain conditions, $\hat{f}$ could generalize well to an unseen sample $\{(x_{n+1}, y_{n+1}), ..., (x_{n+m}, y_{n+m})\}$ drawn from the same domain.

In transfer learning, we introduce a source domain $\mathcal{D}_S = \{\mathcal{X}_S, P_S(X)\}$, a source task $\mathcal{T}_S = \{\mathcal{Y}_S, P_S(Y|X)\}$, a target domain $\mathcal{D}_T = \{\mathcal{X}_T, P_T(X)\}$ and a target task $\mathcal{T}_T = \{\mathcal{Y}_T, P_T(Y|X)\}$. Transfer Learning assumes a difference in the domains (*ie* $\mathcal{D}_S \neq \mathcal{D}_T$) or/and in the tasks (*ie* $\mathcal{T}_S \neq \mathcal{T}_T$). Hence the generalization capacities of traditional machine learning do not hold anymore.

Different taxonomies exist for transfer learning: for instance, in a survey [138], authors propose an assumption-based taxonomy and a data-based taxonomy. Here we propose a taxonomy based on the assumptions made on the shift between source and target dataset. The assumptions can be made on a difference between the probabilities (homogeneous transfer learning) or the input and label spaces (heterogeneous transfer learning):

- $P_S(X) \neq P_T(X)$ corresponds to **domain adaptation**. This is also referenced as sample bias in the literature. Namely, the source and target marginal distributions of the input data are different, hence a model learnt on the source data might not generalize well to data drawn from a different distributions. For instance, one could imagine having seen the consumption of only people living alone and wanting to transfer the knowledge to family houses.

- $P_S(Y_S|X_S) \neq P_T(Y_T|X_T)$ corresponds to **concept shift**. Here, the conditional distributions are different. For instance, if the source is consumption from winter data and the target the consumption from summer data and

one wants to predict the consumption of a heater, there is a shift in the conditional (and also in the marginals, as it often happens in practice). We will also refer to this scenario as **semi-supervised domain adaptation**.

- $\mathcal{X}_S \neq \mathcal{X}_T$ corresponds to **heterogeneous transfer learning**. This could be the case when transferring information from a system monitored by a set of sensors to another system with a different set of sensors.
- $\mathcal{Y}_S \neq \mathcal{Y}_T$ corresponds to a **label set mismatch**. For instance, after learning to predict the consumption of a fridge from the house consumption, one could capitalize on this model to predict the consumption of an oven.

Several of those assumptions may happen at the same time. The more assumptions are broken, the harder the transfer is. Hence, most methods focus on one or two assumptions. Another common assumption that is often mentioned is a **target shift**, *ie* the case when $P_S(Y_S) \neq P_T(Y_T)$.

In our work, unless otherwise stated, we focus on **homogeneous transfer learning** *ie* the input space is the same $\mathcal{X}_S = \mathcal{X}_T$ and the label space is the same $\mathcal{Y}_S = \mathcal{Y}_T$. Hence, we are interested on differences between marginal and conditional distributions on the data. Namely, we are interested in problems where we have a large labeled source dataset and a target dataset with limited or no labeled data but possibly large unlabeled data.

The next sections review the theoretical guarantees of domain adaptation and present an inventory of existing methods. Finally, we review the adaptation with multiple source domains.

2.2 Theory of Domain Adaptation

The intuition behind Domain Adaptation is that it is possible to learn a model on a source domain and transfer it to a target domain if the domains are related. Defining rigorously how both domains should be related for adaptation to be possible is the inspiration of different works on domain adaptation theory. Namely, we seek to bound the error made on target set with information based on the source dataset. As there is a shift in the marginal or the conditional distributions, there is a need to be able to measure the shift between distributions.

Like for transfer learning, Domain Adaptation considers different scenarios, whose name appeared to be used interchangeably in the literature. Hence, in this work we refer to the different scenarios as:

- **Unsupervised Domain Adaptation:** when one has access to labeled source data and unlabeled target data
- **Semi-supervised Domain Adaptation:** when one has access to labeled source data and labeled target data, with possibly unlabeled target data
- **Unsupervised Multi-Source Domain Adaptation:** when one has access to labeled source data coming from different source domains, and unlabeled target data
- **Semi-Supervised Multi-Source Domain Adaptation:** when one has access to labeled source data coming from different source domains, labeled target data, with possibly unlabeled target data

Numerous distances have been introduced to compare probability distributions. They are key to many applications, such as statistical tests, estimation, generative modeling ... In Domain Adaptation, they are used as a tool to compare two domains, and as a quantity to minimize in order to bring domains closer. While classical distances such as Maximum Mean Discrepancy [79], Wasserstein Distance [156] or Total Variation Distance have been used for bounds in Domain Adaptation, the first works on the topic involved distance created specifically for the problem.

We now shortly recall the main generalization bounds based on Vapnik Chervonenkis (VC)-Dimension and Rademacher complexity. Then we present an overview of the bounds based on the divergence-like distance. and finally present other results based on more classical statistical distances.

2.2.1 Generalization bounds

Having theoretical guarantees on the performance and generalization abilities of a model is of primordial importance to machine learning. It has been well studied for traditional machine learning with different approaches. Two of the most common are based on the VC dimension and the Rademacher complexity. We first recall those two main results from statistical learning theory.

Generalization bound with VC-Dimension

In this section, we consider an input set $\mathcal{X}$ and a target set $\mathcal{Y} = \{+1, -1\}$ (binary classification) and an i.i.d. sample $\mathcal{S} = \{(x_1, y_1), ..., (x_n, y_n)\}$ such that $(x_i, y_i) \sim P(X, Y)$. We consider a hypothesis class $\mathcal{H} : \mathcal{X} \to \mathcal{Y}$. For any hypothesis function $h \in \mathcal{H}$, its associated risk is defined as:

$$\mathbf{R}(h) = \mathbb{E}_{(x,y)\sim P(X,Y)}[L(h(x), y))] \tag{2.1}$$

where $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a loss function (for instance $L : (y, y') \to \mathbf{1}(y = y')$). Similarly, its empirical risk is defined as

$$\widehat{\mathbf{R}_{\mathcal{S}}}(h) = \frac{1}{n}\sum_{i=1}^{n} L(h(x_i), y_i) \tag{2.2}$$

A natural question that comes to mind is to explore the error provided by the empirical risk estimation. For this purpose, Vapnik and Chervonenkis introduced the Vapnik-Chervonenkis dimension of an hypothesis class [176]. It is based on the ability of a hypothesis to shatter a set. To define the VC-dimension, one firstly needs to define the growth function, also called the shatter coefficient.

Definition 1. *The growth function $\Pi_{\mathcal{H}}$ of an hypothesis class $\mathcal{H}$ is defined as*

$$\Pi_{\mathcal{H}}(m) = \max_{\{x_1,...,x_m\}\subset\mathcal{X}} |\{(h(x_1), ..., h(x_m)), h \in \mathcal{H}\}| \tag{2.3}$$

where for any set A, $|A|$ is the cardinality of A.

The growth function measures the ability of a hypothesis class to classify m samples in a different way. From this definition, it is possible to define the VC-Dimension. Namely, the VC-Dimension of a hypothesis class $\mathcal{H}$ is the maximum number of samples that an hypothesis class $\mathcal{H}$ is able to shatter completely, *ie* with hypothesis in $\mathcal{H}$, it is possible to assign any labels to the sample $\{x_1, ..., x_m\}$.

Definition 2. *The VC-Dimension of an hypothesis class $\mathcal{H}$ is defined as*

$$VC_{\mathcal{H}} = \max_m \{m | \Pi_{\mathcal{H}}(m) = 2^m\} \tag{2.4}$$

Using this definition, the following generalization bound on the empirical risk was proven [176]:

Theorem 1. *Using the previous notations with $d = VC_{\mathcal{H}}$, for any hypothesis $h \in \mathcal{H}$, for any $\delta \in]0, 1[$, with probability at least $1 - \delta$ over the choice of sample $\mathcal{S} = \{(x_1, y_1), ..., (x_m, y_m)\} \sim (P(X, Y))^m$,*

$$\mathbf{R}(h) \leq \widehat{\mathbf{R}_{\mathcal{S}}}(h) + \sqrt{\frac{4}{m}\left(d \ln \frac{2em}{d} + \ln \frac{4}{\delta}\right)} \tag{2.5}$$

This bound offers an upper-bound on the gap between the empirical estimate of the error and its true value for a sample size m. VC-Dimension has been studied for many well-known classes of hypothesis and is still largely used in statistical learning theory. Bounds for the regression case can also be derived using pseudo-dimension [131].

Generalization bound with Rademacher Complexity

Rademacher Complexity has been introduced as an alternative measure of the complexity of an hypothesis class. While VC-Dimension measures the ability of an hypothesis class to shatter a dataset, Rademacher Complexity intuitively measures its ability to fit to random noise.

Keeping the same notations as before, we introduce the Rademacher variables σ taking binary values in $\{-1, +1\}$ such that $p_\sigma(\sigma = 1) = \frac{1}{2}$ and $p_\sigma(\sigma = -1) = \frac{1}{2}$. Then we introduce the Rademacher Complexity in the next definition.

Definition 3. *For a sample $\mathcal{S} = \{x_1, ..., x_m\}$, the empirical Rademacher complexity of a hypothesis class $\mathcal{H}$ is defined as*

$$\mathcal{R}_{\mathcal{S}}(\mathcal{H}) = \mathbb{E}_\sigma \left[\sup_{h \in \mathcal{H}} \frac{2}{m} \sum_{i=1}^{m} \sigma_i h(x_i)\right] \tag{2.6}$$

where $(\sigma_1, ..., \sigma_m)$ are m i.i.d Rademacher variables. Moreover, the Rademacher complexity of an hypothesis class $\mathcal{H}$ is defined as

$$\mathcal{R}_m(\mathcal{H}) = \mathop{\mathbb{E}}_{\mathcal{S} \sim (P(X))^m} [\mathcal{R}_{\mathcal{S}}(\mathcal{H})] \tag{2.7}$$

The Rademacher Complexity is defined as an expectation over every sample possible when VC-Dimension includes a supremum over the samples. From there, it is possible to derive a bound based on the Rademacher Complexity of the hypothesis class.

Theorem 2. *Using the previous notations , for any hypothesis $h \in \mathcal{H}$, for any $\delta \in [0, 1[$, with probability at least $1 - \delta$ over the choice of sample $\mathcal{S} = \{x_1, ..., x_m\} \sim (P(X, Y))^m$,*

$$\mathbf{R}(h) \leq \widehat{\mathbf{R}_{\mathcal{S}}}(h) + \mathcal{R}_m(\mathcal{H}) + \sqrt{\frac{\ln(1/\delta)}{2m}} \tag{2.8}$$

Rademacher complexity has been studied for many methods, such as SVM, Neural Networks, ... The Rademacher complexity is data dependent and leads to tight bounds. Moreover, it has been extended to regression in an easier way than through pseudo VC-Dimensions. In the regression case, $\mathcal{Y} \subset \mathbb{R}$. The following generalization bound holds in the regression case [131]:

Theorem 3. *Using the previous notations, let $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ be bounded by $M > 0$, ie $\sup_{y,y' \in \mathcal{Y}} L(y, y') \leq M$. Moreover, for any $y' \in \mathcal{Y}$, we assume $y \to L(y, y')$ to be μ-Lipschitz. Then for any hypothesis $h \in \mathcal{H}$, for any $\delta \in]0, 1[$, with probability at least $1 - \delta$ over the choice of sample $\mathcal{S} = \{(x_1, y_1), ..., (x_m, y_m)\} \sim (P(X, Y))^m$,*

$$\mathbf{R}(h) \leq \widehat{\mathbf{R}_{\mathcal{S}}}(h) + 2\mu\mathcal{R}_m(\mathcal{H}) + M\sqrt{\frac{\ln 1/\delta}{2m}} \tag{2.9}$$

For instance, if we assume $\mathcal{Y}$ to be a bounded subset of $\mathbb{R}$, then the ℓ^2 loss $L(y, y') = (y - y')^2$ respects the assumptions of the theorem.

Other generalization bounds have been proposed based on PAC-Bayesian Theory [126] or Stability [33] but are out of the scope of this work. Now that we have introduced some of the classical generalization bounds in the typical machine learning setting, we extend them in a presence of a shift in the distributions.

2.2.2 Divergence-based Domain Adaptation

The previous framework focuses on the generalization ability to a sample drawn from the same distribution. In this part, we review bounds for a sample drawn from a different distribution presented in [25] and [121].

$\mathcal{H}$-Divergence based Domain Adaptation [25]

In the seminal paper of Ben-David et al. (2006) [24], authors define a theoretical framework for domain adaptation introduced a specific distance and obtained a bound that was later corrected in [30] and [25]. They consider the framework of domain adaptation with two domains $\mathcal{D}_{\mathcal{S}} = \{\mathcal{X}, p_S(X_S)\}$ and $\mathcal{D}_{\mathcal{T}} = \{\mathcal{X}, p_T(X_T)\}$ and labeling

functions f_S and f_T such that $f_T : \mathcal{X} \to \{0,1\}$ and $f_T : \mathcal{X} \to \{0,1\}$. They consider an hypothesis class $\mathcal{H} : \mathcal{X} \to \{0,1\}$ and the loss function $L(y,y') = |y - y'|$. Then for any hypothesis $h \in \mathcal{H}$, the source risk is defined as $\mathbf{R}_S(h) = \mathop{\mathbb{E}}_{x\sim p_S}[|h(x) - f_S(x)|]$ and target risk as $\mathbf{R}_T(h) = \mathop{\mathbb{E}}_{x\sim p_T}[|h(x) - f_T(x)|]$.

The first bound they propose is based on the Total Variation distance defined as follows:

Definition 4. *For two probability measures P and Q defined over a set of measurable subsets $\mathcal{B}$, the total variation distance d_{TV} is defined as*

$$d_{TV}(P,Q) = 2 \sup_{B\in\mathcal{B}} |P(B) - Q(B)| \tag{2.10}$$

Then the following bound holds:.

Theorem 4. *For any hypothesis $h \in \mathcal{H}$,*

$$\mathbf{R}_T(h) \leq \mathbf{R}_S(h) + d_{TV}(p_S, p_T) + \lambda \tag{2.11}$$

where $\lambda = \min\left[\mathop{\mathbb{E}}_{x\sim p_S}[|f_S(x) - f_T(x)|], \mathop{\mathbb{E}}_{x\sim p_T}[|f_S(x) - f_T(x)|]\right]$

This bound relates the target risk to the source risk, a distance between marginal distributions and a distance between conditional distributions (in the form of labelling functions). This bound provides the first keys for domain adaptation and gives ground to adaptation models that try to find representations bringing the domains closer. The main issue is that the total variation distance is not possible to estimate it with finite samples, hence impossible to use in practice as a criterion for adaptation. Moreover, it is a supremum over every possible subset, which means that the bound is very loose.

Based on these remarks, Ben-David et al. (2010) [25] propose the $d_{\mathcal{H}}$ distance, inspired from [101]:

Definition 5. *For two probability measures P and Q defined over a set $\mathcal{X}$ and an hypothesis class $\mathcal{H}$, and noting $I : h \to \{x \in \mathcal{X} : h(x) = 1\}$, the $\mathcal{H}$-divergence is defined as*

$$d_{\mathcal{H}}(P,Q) = 2 \sup_{h\in\mathcal{H}} |P(I(h)) - Q(I(h))| \tag{2.12}$$

This divergence is always smaller than d_{TV}, depends directly on the hypothesis class $\mathcal{H}$ and can be estimated from finite samples. Indeed, if we note $\mathcal{S}_S$ and $\mathcal{S}_T$ two samples of size m from P_S and P_T, the empirical estimation of $d_{\mathcal{H}}$ is

$$\hat{d_{\mathcal{H}}}(\mathcal{S}_S, \mathcal{S}_T) = 2\left(1 - \min_{h\in\mathcal{H}}\left[\frac{1}{m}\sum_{x:h(x)=0} I(x \in \mathcal{S}_S) + \frac{1}{m}\sum_{x:h(x)=1} I(x \in \mathcal{S}_T)\right]\right) \tag{2.13}$$

So computing the $d_{\mathcal{H}}$ between two domains boils down to finding a classifier able to discriminate between domains. Moreover, the empirical estimation of $d_{\mathcal{H}}$ converges towards its true value if $\mathcal{H}$ has finite VC-Dimension

[101] with probability $1-\delta$ over the samples of $\mathcal{S}_S$ and $\mathcal{S}_T$:

$$d_{\mathcal{H}}(p_S, p_T) \leq \hat{d_{\mathcal{H}}}(\mathcal{S}_S, \mathcal{S}_T) + 4\sqrt{\frac{2VC_{\mathcal{H}}\log(2m) + \log(\frac{2}{\delta})}{m}} \tag{2.14}$$

To define a bound between source and error risks, the symmetric difference hypothesis space $\mathcal{H}\Delta\mathcal{H}$ is introduced as follows:

Definition 6. *For an hypothesis space $\mathcal{H}$, the symmetric difference hypothesis space $\mathcal{H}\Delta\mathcal{H}$ is defined as*

$$\mathcal{H}\Delta\mathcal{H} = \{g : x, x' \in \mathcal{X} \rightarrow h(x) \oplus h'(x); h, h' \in \mathcal{H}\} \tag{2.15}$$

where $\oplus$ is the XOR operator.

Finally, the following bound holds using this theory:

Theorem 5. *Using the previous notations, for any hypothesis $h \in \mathcal{H}$,*

$$\mathbf{R}_T(h) \leq \mathbf{R}_S(h) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(p_S, p_T) + \lambda \tag{2.16}$$

where $\lambda = \min_{h\in\mathcal{H}} [\mathbf{R}_S(h) + \mathbf{R}_T(h)]$

This bound is similar to Equation 2.11 as the target risk is bounded by the source risk, a divergence between marginal probabilities and a term on ideal hypothesis for both domains. $\mathbf{R}_S(h)$ can be estimated from finite samples as shown with typical generalization bounds. $d_{\mathcal{H}\Delta\mathcal{H}}$ can also be estimated using Equation 2.13 and the fact that the $VC_{\mathcal{H}\Delta\mathcal{H}} \leq 2VC_{\mathcal{H}}$. Finally, the last term cannot be estimated without labels or further assumptions, such as the structure of the labelling functions. Thus it is assumed to be small for unsupervised domain adaptation. When the labelling functions are too different, one does not have guarantees of good adaptation, which can lead to negative transfer.

One can notice that the bound involves the symmetric difference hypothesis class $\mathcal{H}\Delta\mathcal{H}$ and not the original $\mathcal{H}$, and in practice, the computation of $d_{\mathcal{H}\Delta\mathcal{H}}$ is intractable. Hence, the authors of [25] suggest to approximate it by training a classifier from $\mathcal{H}$ to discriminate between domains, *ie* approximate $d_{\mathcal{H}}$ which only is a lower bound of $d_{\mathcal{H}\Delta\mathcal{H}}$.

Different extensions have been made to this bound:

- Semi-supervised Domain Adaptation: Ben-David et al. (2010) [25] tackle the scenario where labeled target data is available. They derive a bound mixing labeled source and target data, which overcomes the assumption made on λ in Equation 2.16. A very interesting result is that they derive the number of target labeled samples from which there are enough target samples to learn completely, *ie* the adaptation becomes useless

- Multi-Source Domain Adaptation: Ben-David et al. (2010) [25] and [196] tackle the issue of adapting from different source domains. They derive a bound based on a convex combination of source distributions to adapt to the target.

The seminal work of Ben-David et al. (2010) [25] gave the first formal description of domain adaptation for binary classification. It can be extended to multi-class classification, for instance by decomposing the problem into binary classification. But the extension to regression problems is not straightforward and is the topic of the next section.

Discrepancy based Domain Adaptation

The previous bounds only hold for classification so the applications are limited. Hence, authors of [121] developed a more general framework. In their framework, they consider general output space $\mathcal{Y}$ and loss functions $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$. In the following, for two hypotheses $h, h' \in \mathcal{H}$, we note $\mathbb{R}_S(h, h') \underset{x\sim p_S}{\mathbb{E}}[L(h(x), h'(x))]$ and $\mathbb{R}_T(h, h') \underset{x\sim p_T}{\mathbb{E}}[L(h(x), h'(x))]$. The source and target risks of an hypothesis $h \in \mathcal{H}$ are defined as $\mathbf{R_S}(h) = \mathbf{R_S}(h, f_S)$ and $\mathbf{R_T}(h) = \mathbf{R_T}(h, f_T)$

Authors introduce the discrepancy distance, which is similar to the $d_{\mathcal{H}}$ divergence.

Definition 7. *For two probability measures P and Q defined over a set $\mathcal{X}$, an hypothesis class $\mathcal{H} : \mathcal{X} \to \mathcal{Y}$ and a loss $L :: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$, the discrepancy between P and Q is defined as*

$$\textit{disc}_{\mathcal{H},L}(P, Q) = \sup_{h,h'\in\mathcal{H}} \left| \underset{x\sim p_S}{\mathbb{E}}[L(h(x), h'(x))] - \underset{x\sim p_T}{\mathbb{E}}[L(h(x), h'(x))] \right| \tag{2.17}$$

It is closely related to $d_{\mathcal{H}\Delta\mathcal{H}}$ as for $L^1 : (y, y') \to |y - y'|$, they are related by $disc_{\mathcal{H},L^1} = \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}$. So the discrepancy can be seen as a generalization of the $d_{\mathcal{H}\Delta\mathcal{H}}$ to other classes of problems. For this general class of problems, most bounds are provided using Rademacher complexity rather than VC-Dimension.

Theorem 6. *Using the previous notations, let $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ such that $L(y, y') = |y - y'|^q$ be bounded by M, ie $\sup_{y,y'\in\mathcal{Y}} L(y, y') \leq M$. Then with probability $1 - \delta$ over the sampling of $\mathcal{S}_S$ and $\mathcal{S}_T$ of size m from p_s and p_t, we have*

$$\textit{disc}_{\mathcal{H},L}(p_s, p_t) \leq \widehat{\textit{disc}}_{\mathcal{H},L}(\mathcal{S}_S, \mathcal{S}_T) + 4q(\mathcal{R}_S(\mathcal{H}) + \mathcal{R}_T(\mathcal{H})) + 3M\sqrt{2\frac{\log\frac{2}{\delta}}{m}} \tag{2.18}$$

This bound, presented for ℓ^q-losses also holds for any bounded loss by modifying the class of the Rademacher complexity to a composition with the loss. Similarly to the bound 2.16, one can derive the following general bound using the discrepancy

Theorem 7. *Using the previous notations and the assumptions of Theorem 6, noting $h_S^* = \arg\min_{h\in\mathcal{H}} \mathcal{R}_S(h)$ and*

$h_T^* = \arg\min_{h \in \mathcal{H}} \mathcal{R}_T(h)$, *we have*

$$\mathcal{R}_T(h) \leq \mathcal{R}_S(h, h_S^*) + \textit{disc}_{\mathcal{H},L}(p_S, p_T) + \lambda \tag{2.19}$$

where $\lambda = \mathcal{R}_S(h_T^*) + \mathcal{R}_T(h_S^*, h_T^*)$

Once again, this bounds connects the target risk with the source risk, the discrepancy between marginal distributions and a term connecting the true labelling functions. It is not directly comparable to Equation 2.16 as it involves ideal hypotheses h_S^* and h_T^*, but for instance, if one assumes that $h_S^* = h_T^*$, then the bound based on the discrepancy becomes tighter.

The main contribution of this bound is the generalization to a larger class of loss, and hence problems. In the original paper, authors proposed a domain adaptation method for binary classification and regression based on kernels. Different follow-up works show bounds for specific class of predictors, mainly kernels [44] [46].

Extensions have been proposed to tackle other scenarios than unsupervised domain adaptation:

- Semi-supervised Domain Adaptation: authors from [44] [46] introduced the generalized discrepancy and the $\mathcal{Y}$-discrepancy to allow the use of labeled target data and derive bounds similar to Equation 2.19
- Multi-Source Domain Adaptation: during this PhD, we developed a bound and a method for adaptation from multiple sources [152]

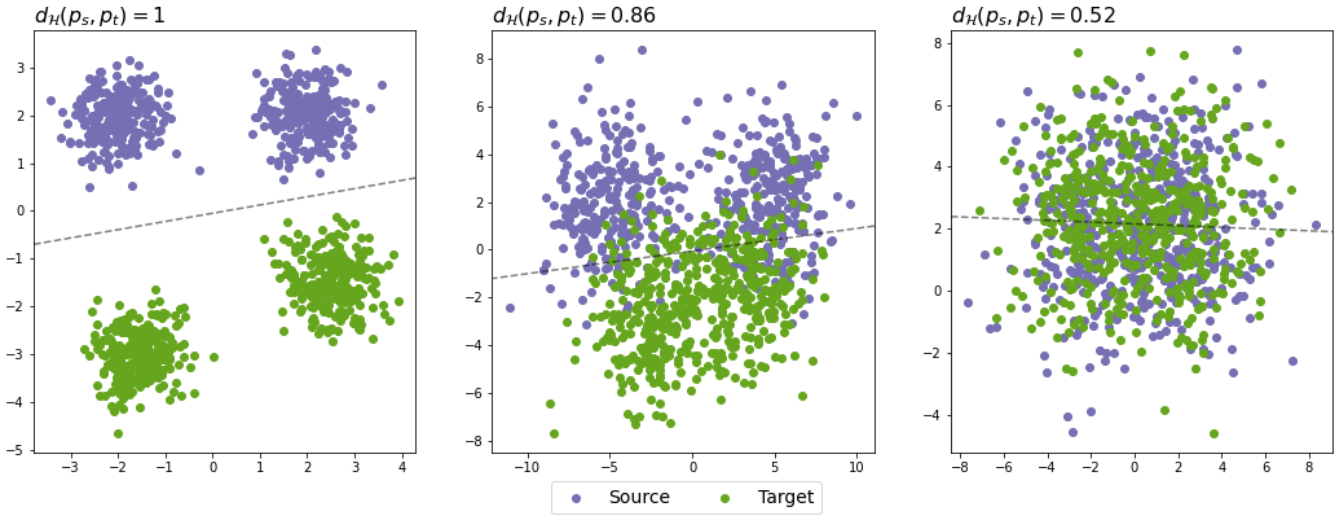


Figure 2.2: Illustration of the $\mathcal{H}$-divergence with linear classifiers

Those two divergences are the main results of domain adaptation theory. They both base the divergence between domains depending on the hypothesis class. In Figure 2.2 we show the $d_{\mathcal{H}\Delta\mathcal{H}}$ for different domains for linear classifier class $\mathcal{H}$. One can note that using a more complex class of hypotheses, $d_{\mathcal{H}\Delta\mathcal{H}}$ would be smaller for the

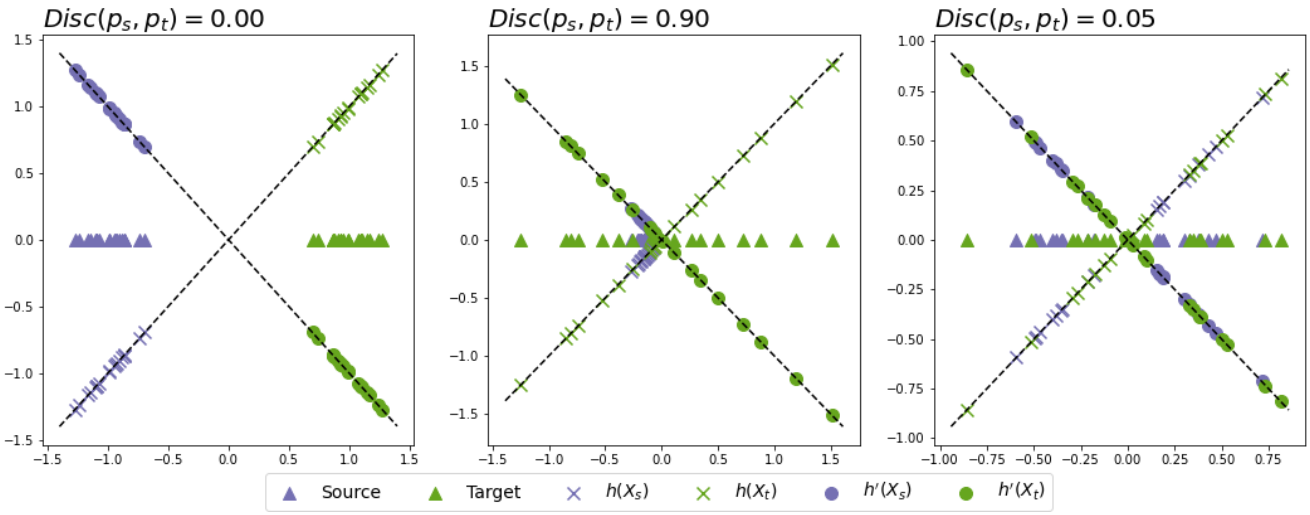


Figure 2.3: Illustration of the discrepancy with linear regressors $\mathcal{H} = \{h : x \to w^T x \ ; \ \|w\|_2 \leq 1\}$ and ℓ^2-loss. Source and target data are generated as 1D-Gaussian distributions centered on -1 and $+1$ with different standard deviations in each graph ($\{1, 1\}$, $\{0.5, 5\}$ and $\{4, 4\}$).

second example. Hence there is a tradeoff between hypothesis class complexity to potentially have a smaller true value $d_{\mathcal{H}\Delta\mathcal{H}}$ and the generalization ability of its estimation, as in classical machine learning.

In Figure 2.3, we illustrate the Discrepancy with bounded linear regressors on 1-dimensional data. One can already observe the difficulty of domain adaptation in regression, as the label space is a subset of $\mathbb{R}$ and not finite, leading to possibly very large discrepancies. More details about the difference between classification and discrepancy are provided in Chapter 4.

2.2.3 Alternative approaches

The previous sections reviewed domain adaptation theory with tailored divergence. More traditional statistical distances have been proposed for domain adaptation, especially the Maximum Mean Discrepancy and the Wasserstein Distance. We only present the framework for Maximum Mean Discrepancy and give an overview of other frameworks as they are less related to our work.

Maximum Mean Discrepancy based Domain Adaptation

First we define the Maximum Mean Discrepancy.

Definition 8. *Let $\mathcal{H}_k$ be a Reproducing Kernel Hilbert Space over a set $\mathcal{X}$ with associated kernel k. For two distributions defined over $\mathcal{X}$, The Maximum Mean Discrepancy is defined as*

$$MMD(P, Q; \mathcal{H}_k) = \sup_{f \in \mathcal{H}_k} \left| \mathop{\mathbb{E}}_{x \sim p_s} [f(x)] - \mathop{\mathbb{E}}_{x \sim p_t} [f(x)] \right| \tag{2.20}$$

Maximum Mean Discrepancy is widely used as under the condition that the kernel is universal, every moment of the distributions match. Moreover, an unbiased estimate from two samples $\mathcal{S}_S = \{x_S^{(1)}, ..., x_S^{(n_S)}\}$ and $\mathcal{S}_T = \{x_T^{(1)}, ..., x_T^{(n_T)}\}$ can be derived as

$$\widehat{MMD}(\mathcal{S}_S, \mathcal{S}_T; \mathcal{H}_k)^2 = \frac{1}{n_S^2} \sum_{i,j=1}^{n_S} k(x_S^{(i)}, x_S^{(j)}) + \frac{1}{n_T^2} \sum_{i,j=1}^{n_T} k(x_T^{(i)}, x_T^{(j)}) - \frac{2}{n_S n_T} \sum_{i=1}^{n_S} \sum_{j=1}^{n_T} k(x_S^{(i)}, x_T^{(j)}) \tag{2.21}$$

A common way to write this estimation is $\widehat{MMD}(\mathcal{S}_S, \mathcal{S}_T; \mathcal{H}_k)^2 = Tr(KL)$ where $K \in \mathbf{R}^{(n_S+n_T)\times(n_S+n_T)}$ is the kernel matrix induced by k on $\mathcal{S}_S \cup \mathcal{S}_T$ and L is defined by $L_{ij} = \frac{1}{n_S^2}$ if $x_i, x_j \in \mathcal{S}_S$; $L_{ij} = \frac{1}{n_T^2}$ if $x_i, x_j \in \mathcal{S}_S$ and $L_{ij} = -\frac{1}{n_S n_T}$ otherwise.

With this metric, it is possible to derive a bound with a similar interpretation to the one based on divergence [94]. This bound holds for losses $L : (y, y') \to |y - y'|^q$ for $q > 0$.

Theorem 8. *Using previous notations, let $\mathcal{H} = \{f \in \mathcal{H}_k \; ; \; \|f\|_2 \leq 1\}$. If L respects the triangle inequality and $\|l\|_{\mathcal{H}_k^q} \leq 1$, with probability $1 - \delta$ over the samples $\mathcal{S}_S$ and $\mathcal{S}_T$, we have for all $h \in \mathcal{H}$*

$$\mathbf{R}_T(h) \leq \mathbf{R}_S(h) + \widehat{MMD}(\mathcal{S}_S, \mathcal{S}_T) + \lambda + 2\sqrt{\frac{\log \frac{2}{\delta}}{2m}} + \frac{2}{m} \left(\mathop{\mathbb{E}}_{\mathcal{S}\sim p_s} [\sqrt{Tr(K_{\mathcal{S}})}] + \mathop{\mathbb{E}}_{\mathcal{T}\sim p_s} [\sqrt{Tr(K_{\mathcal{T}})}] \right) \tag{2.22}$$

where $\lambda = \min_{h\in\mathcal{H}} [\mathbf{R}_S(h) + \mathbf{R}_T(h)]$.

Once again, the bound involves the source risk, the Maximum Mean Discrepancy between marginal distributions, a term λ between labels and some complexity terms. The main interest of using the MMD instead of the divergences lies in its easy estimation explaining why many methods are based on the MMD. MMD-based domain adaptation also has been extended to other scenarios such as semi-supervised learning.

Other Frameworks

Bounds with Wasserstein distance inspired from Optimal Transport [179] have also been introduced [148] in the same spirit as the ones from Maximum Mean Discrepancy as Wasserstein distance also is an Integral Probability Metric. Indeed, for $\mathcal{H} = h : \|h\|_L \leq 1$ a set of Lipschitz-1 hypotheses, the Wasserstein distance can be computed as

$$W_1(P, Q) = \sup_{h; \|h\|_L \leq 1} |\mathbb{E}_{x\sim P}[h(x)] - \mathbb{E}_{x\sim Q}[h(x)]| \tag{2.23}$$

A new line of work on PAC-Bayesian Domain Adaptation [73] [74] appeared. It is particularly promising as the bounds involve terms on the expected shift between domains instead of a supremum as in the $d_{\mathcal{H}}$ or the Discrepancy. Namely, the bounds are tighter than the ones of Ben-David et al. (2010) [25], although only algorithms for linear classifiers have been derived. Using Bayesian theory, it is also possible to encode some preliminary knowledge about the relationships between domains in this new measure. Another popular measure of discrepancy between

domains has been the Rényi divergence, especially for domain adaptation with multiple sources [88] [123]. For more details about those methods, we refer the interested reader to the book of Redko et al. (2019) [149].

2.2.4 Summary

Bounds for domain adaptation have been proposed under several frameworks and share some similarities. In order to be able to generalize to a target domain, we need to learn an hypothesis that can

- Minimize the source risk
- Minimize a divergence between marginal probabilities
- If labels are unavalaible, assume that the ideal hypotheses on each domains are close. Otherwise, use them to control the divergence between conditional probabilities.

Both divergence measures ($d_{\mathcal{H}\Delta\mathcal{H}}$ and its extension $Disc$) are nice tools for domain adaptation as they include the problem at hand in the bound, and in the design of algorithms. But as they respectively involve a supremum over $\mathcal{H}\Delta\mathcal{H}$ or over two hypotheses, they require either an approximation (with $d_{\mathcal{H}}$ for $d_{\mathcal{H}\Delta\mathcal{H}}$) or a limitation in the hypothesis space (to linear or kernelized regressors for instance) to be estimated.

Metrics such as the Maximum Mean Discrepancy do not suffer from the same issue. For this reason, many methods are based on the MMD, even before the theoretical bounds were shown. But MMD is restricted to kernels, which can be an issue for some applications. In the end, the choice of the metric for adaptation depends on the nature of the problem.

In Chapter 4, we derive bounds based on a novel discrepancy measure between distributions. Similarly to the original discrepancy, our bounds hold for both regression and classification problems. Moreover, we derive an efficient way to integrate the discrepancy in the training of neural networks.

2.3 Existing approaches for Homogeneous Transfer Learning

The bounds seen in the previous section gave the intuition behind what quantities a method for domain adaptation should try to control. Naturally, the question becomes how to use those bounds and what knowledge should one transfer from the source to the target. Consider a fault detection problem with a source dataset made of several wind plants monitored with some sensors that we want to transfer to a new target wind plant. In some cases, only a few of the source wind plants operate in a similar environment to the target one. Then we would want to give a higher weight to these specific source *instances*. Or it could be that the source and target datasets only share the same distributions over some sensors. Then we would want to extract new *features* to align the domains. Finally,

without explicit assumptions on the nature of the transfer, one could use the source *model parameters* as an input for the target model, either as initial parameters or as constraints. To summarize, we differentiate three categories:

- Instance-based domain adaptation: where an algorithm tries to select or re-weight the source data points based on their relevance for adaptation
- Feature-based domain adaptation: where a feature transformation is performed on the source or target domains to bring the domain closer
- Model-based domain adaptation: where the parameters of a model learnt on the source data are transferred to the target model

For instance, the authors who proved the bound based on $d_{\mathcal{H}\Delta\mathcal{H}}$ based their algorithm for feature-based domain adaptation whereas the authors of the seminal paper on discrepancy [121] based their algorithm on instance re-weighting. Some methods are hybrid as they perform two of these adaptations at the same time.

2.3.1 Instance-based domain adaptation

Broadly speaking, instance-based domain methods aim to select or re-weight the source instances depending on their relationship to the target. Namely, in terms of distributions, if we consider two domains $\mathcal{D}_S = \{\mathcal{X}, P_S(X_S)\}$ and $\mathcal{D}_T = \{\mathcal{X}, P_T(X_T)\}$ with respective tasks $\mathcal{T}_S = \{\mathcal{Y}, P_S(Y_S|X_S)\}$ and $\mathcal{T}_T = \{\mathcal{Y}, P_T(Y_T|X_T)\}$, the aim of those methods is to find a weighting function w such that, for $x \in \mathcal{X}$, $P_T(x) = w(x)P_S(x)$ for unsupervised domain adaptation or $P_T(y|x) = w(x)P_S(y|x)$ for semi-supervised domain adaptation.

One can note that, under the assumption that $P_S(y|x) = P_T(y|x)$, the target risk can be re-written as:

$$\mathbf{R}_T(h) = \mathbb{E}_{(x,y)\sim P_T}[L(h(x), y)] = \mathbb{E}_{(x,y)\sim P_S}\left[\frac{P_T(x,y)}{P_S(x,y)}L(h(x), y)\right] = \mathbb{E}_{(x,y)\sim P_S}\left[\frac{P_T(x)}{P_S(x)}L(h(x), y)\right] \tag{2.24}$$

A straightforward solution is to directly estimate $P_S(x)$ and $P_T(x)$, and use $w(x) = \frac{P_T(x)}{P_S(x)}$ as weights for the source data, as was proposed in [192] [111]. This solution has two drawbacks: firstly, it requires that both distributions have the same support, and small errors on low values of $P_S(x)$ gives high weights to undesire data points. Secondly, the estimation of $P_T(x)$ may require a large number of points and is not related to the final prediction task. It is even more the case for semi-supervised learning where estimating $P_T(Y_T|X_T)$ is as complicated as learning a predictor h_T. Encoding this prediction task in the estimation of the weights may lead to more efficient methods.

Unsupervised domain adaptation

Discrepancy-based. Hence Mansour et al. (2009) [121] presented a re-weighting algorithm for linear regressors with ℓ^2-loss. Namely, they consider a labeled source sample $\mathcal{S}_S = \{(x_S^{(1)}, y_S^{(1)}), ..., (x_S^{(n_s)}, y_S^{(n_S)})\}$ and an unlabeled target sample $\mathcal{S}_T = \{x_S^{(1)}, ..., x_T^{(n_T)}\}$ with an empirical distributions $\widehat{p_S} = \frac{1}{n_S}\sum_{i=1}^{n_S} \mathbf{1}_{x_S^{(i)}}$ and $\widehat{p_T} = \frac{1}{n_T}\sum_{i=1}^{n_T} \mathbf{1}_{x_T^{(i)}}$. They define the re-weighted empirical source distribution as $\widehat{p_S}' = \sum_{i=1}^{n_S} \beta_i \mathbf{1}_{x_S^{(i)}}$. Then the objective formulated to compute the weights β with the discrepancy is:

$$\min_{\beta, \|\beta\|_1 \leq 1} Disc(\widehat{p_S}', \widehat{p_T}; \mathcal{H}, \ell^2) \tag{2.25}$$

where $\mathcal{H} = \{h : x \to w^T x \,|\, \|w\|_2 \leq 1\}$ and $\ell^2 : (y, y') \to (y - y')^2$. With these choices of hypothesis space and loss it is possible to show that the problem can be solved efficiently using semi-definite programming [121]. Later these results were extended to kernel functions with the generalized discrepancy [46], and using quadratic programming to solve the problem: Generalized Discrepancy Minimization (GDM).

Kernel Mean Matching (KMM). Similarly, Maximum Mean Discrepancy can be used to compare domains. It is proposed in [91] [78] where the formulated objective of KMM is

$$\min_{\beta, \|\beta\|_1 \leq 1} MMD(\widehat{p_S}', \widehat{p_T}; \mathcal{H}_k) \tag{2.26}$$

where k is a chosen universal kernel, typically the Gaussian kernel. With the empirical estimation of the Maximum Mean Discrepancy described in Equation 2.21, the objective can be formulated as

$$\min_{\beta, \|\beta\|_1 \leq 1} \frac{1}{n_S}\beta^T K \beta - \frac{2}{n_S^2}\kappa^T \beta \tag{2.27}$$

where $K_{ij} = k(x_S^{(i)}, x_S^{(j)})$ and $\kappa_{ij} = k(x_S^{(i)}, x_T^{(j)})$ which can be solved as a quadratic program.

Kullback-Leibler Importance Estimation Procedure. The same kind of problem is formulated using Kullback-Leibler divergence [168]. Namely, they assume that the weights can be computed as $w(x) = \sum_{j=1}^{b} \alpha_l \phi_l(x)$ where b and $\{\phi_1, ..., \phi_b\}$ are hyperparameter of a chosen basis of functions. The adaptation problem then is

$$\min_{\alpha} \frac{1}{n_S} KL(\widehat{p_S}' || p_T) \tag{2.28}$$

which can be formulated as

$$\max_{\alpha} \frac{1}{n_S} \log \left(\sum_{l=1}^{b} \alpha_l \phi_l(x_T^{(i)}) \right) \quad \text{subject to} \sum_{i=1}^{n_S} \sum_{l=1}^{b} \alpha_l \phi_l(x_S^{(i)}) = 1 \tag{2.29}$$

This can be solved by gradient descent as the objective is convex.

Discriminative Learning. Bickel et al. (2007) [28] try to model the weights $w(x)$ as the probabilities given by a classifier trained to discriminate between domains. Namely, they learn a logistic regression classifier on source and target data that outputs an estimate of the appartenance of a point to the target dataset. Then the weight given to a source dataset is higher if it is considered likely to belong to the target data. An interesting point is the link that can be made between this method and the $d_{\mathcal{H}}$ distance.

Semi-supervised domain adaptation

Using previous methods. The previous methods can be extended to semi-supervised domain adaptation by computing weights for each class, *ie* replace $P(X_S)$ and $P(X_T)$ by $P(X_S|Y_S)$ and $P(X_T|Y_T)$ in the objectives. But this would require sufficient labeled data, hence, some methods are dedicated to semi-supervised domain adaptation.

TrAdaBoost. In a traditional boosting algorithm, weak learners are learnt sequentially on a re-weighted training samples [199]. At each step, higher weights are given to samples with high error. The TrAdaBoost algorithm [50], uses a similar scheme to re-weight the source samples. Namely, at each step the source samples that are well classified by the current learner are given higher weights. It is possible to show that the expected error made on data drawn from the target domain is not higher than the one made by a learner using only labeled target data. Later it was extend to TrAdaBoostR2 [140] for regression.

Summary

Those methods differ in the criteria on which the weights are computed. The MMD and discrepancy-based methods lead to similar formulations and results. KLIEP suffers from the fact that the Kullback-Leibler divergence is not defined when the support of P_T is not included in the one of P_S. TrAdaBoost is very powerful when labeled target data is available but it is not the case of unsupervised domain adaptation.

In fact, the source and target probability distributions cannot be ”too far” from one another for those methods to perform. For instance, if their support does not overlap, then the weights are very unlikely to have any meaningful

sense for adaptation. As such, the name *sample selection bias* used in [91] is adapted for this framework. We do not expect these methods to work well when the domains are very different from each other.

2.3.2 Feature-based domain adaptation

Feature-based domain adaptation methods are the second large class of domain adaptation methods. They aim to find a feature transformation of the input data that brings the domains closer. Namely, in terms of distributions, if we consider two domains $\mathcal{D}_S = \{\mathcal{X}, P_S(X_S)\}$ and $\mathcal{D}_T = \{\mathcal{X}, P_T(X_T)\}$ with associated tasks $\mathcal{T}_S = \{\mathcal{Y}, P_S(Y_S|X_S)\}$ and $\mathcal{T}_T = \{\mathcal{Y}, P_T(Y_T|X_T)\}$, the aim of those methods is to find a feature extraction function ϕ such that, for $x \in \mathcal{X}$, $P_T(\phi(x)) = P_S(\phi(x))$ (symmetrical feature adaptation) or two functions ϕ and ϕ' such that $P_T(\phi'(x)) = P_S(\phi(x))$ (asymmetrical feature adaptation).

Feature engineering is key to the success of many machine learning applications and feature selection [80] is often a necessary pre-processing step for many learning algorithms. Different types of methods exist for feature-based domain adaptation.

Traditional Methods

Frustratingly Easy Domain Adaptation. One of the first methods [52] makes the assumption that some features are only relevant for the source domain, some only for the target domain and some for both domains. Then, one would want to ideally extract the common features to both domains. To do that, they augment the feature space by duplicating the features and creating artificial source and target features: new variables are written $x'_S = (x_S, 0, x_S)$, $x'_T = (0, x_T, x_T)$. Finally, a linear classifier is learnt based on those new variables. Hence, this method requires labels in the source and target domains.

Moment-matching representation-based domain adaptation. Many methods aim to find a representation that aligns some or all of the moments of the distributions. It is often done using Maximum Mean Discrepancy (MMD), as it has been shown that when using a universal kernel, if the MMD between source domains and target domains is zero, then the domains are perfectly aligned [79].

Transfer Component Analysis [139] proposes to extract features that minimize the MMD between both domains. Namely, they use the empirical estimation of the MMD (Equation 2.21) $MMD(X_S, X_T) = Tr(KL)$ to formulate a kernel learning problem by introducing a parameter $W \in \mathbb{R}^{(n_S+n_T)\times m}$ and the associated kernel $\tilde{K} = KWW^TK$. In order to learn the parameter W, they formulate the objective

$$\min_{W} Tr(W^T W) + \mu Tr(W^T KLKW)$$
$$\text{s.t. } W^T KHKW = I \tag{2.30}$$

where $H = I - 1/(n_S + n_T)1_{n_S+n_T}1^T_{n_S+n_T}$ is a centering matrix. After solving the problem, one expects the representations of source and target domains induced by the kernel $\tilde{k}$ to be aligned. This work was later extended to semi-supervised domain adaptation [113] with Joint Distribution Adaptation. Later the Adaptation Regularization transfer learning [114] framework proposed a unifying view on these methods.

CORrelation ALignment (CORAL) [170] proposes a similar objective by aligning second-order moment of the distributions. Namely, it whitens the source distributions and "re-colors" it with target covariance, which is similar to aligning the distributions with MMD using a polynomial kernel of degree 2. Along with the extension to deep neural networks [169], appears to give good results for many applications as a simple technique.

Subspace-based domain adaptation. Another very common assumption in domain adaptation is that while the representations of the domains in the initial feature space are not aligned, there exists a subspace where source and target data share the same distribution. From this assumption, different methods are proposed.

Geodesic Flow Kernel [76] is proposed as a method to align subspace representation of source and target data. Namely, a PCA is performed onto the source and target data giving two subspaces. Then, using concepts from Riemannian geometry, authors define a kernel to align both subspaces. This method is very dependent on the choice of dimensionality reduction and can be extended to other methods than PCA.

Subspace Alignment [67] follows a similar idea and aims to align two domains. Authors aim to align the subspace obtained from source data to the one obtained from target data. After representations are aligned a classifier is learnt on the projected source data. It is later extended to semi-supervised domain adaptation [68] and refined in [6].

Domain Adaptation with Neural Networks

With the success of representation learning based on neural networks, domain adaptation methods is proposed using neural networks. For instance, some methods based on autoencoders [75] [40] is proposed by simply learning autoencoders on a combination of source and target data.

Domain-adversarial methods is introduced for domain adaptation in [72] and [4]. Authors propose domain adversarial training for neural networks. Taking inspiration from GANs [77] they use a neural network to learn features that are discriminative for the task at hand and for which domains cannot be separated by a classifier.

The way it is done is the following: the first section of the network is used as feature extractor. Then, the network is separated into two branches: a predictor and a discriminator. The predictor learns to classify well the

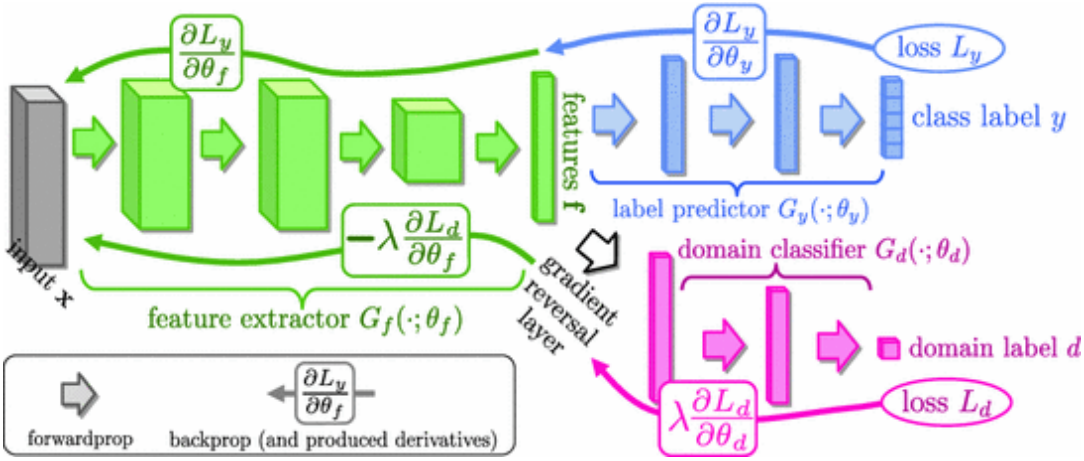


Figure 2.4: Domain Adversarial Neural Network (figure from [72])

source examples whereas the discriminator learns to discriminate between domains. Then the gradient feeded to the feature extractor is reversed, so that it maximizes the discrimination loss between domains.

This method can be linked with the bound based on $d_{\mathcal{H}}$ seen in the previous section. Numerous methods have been proposed for domain-adversarial training of neural networks, including one of the contributions of this thesis. Hence, we give a more extensive discussion and comparison in Chapter 4.

Summary Those methods differ in the criteria chosen to extract common features for both domains. Maximum Mean Discrepancy is a very popular tool as it allows a straightforward use of kernels and an empirical estimation. Divergence-based methods were less popular as their empirical estimation is hard in general. The recent surge of adversarial learning encouraged new methods using this criteria.

For those methods, the source and target datasets can be far from one another on their initial feature representation. Using a projection on a smaller subspace, may it be linear or based on a more complex structure, one expects to be able to find representations that can be discriminative (able to minimize the source risk) and where domains are aligned (able to minimize the difference between domains).

2.3.3 Alternative methods

Here, we mention other popular methods that do not fall in one of the previous categories.

Optimal Transport. Optimal Transport is a long-standing problem in mathematics finding applications in many fields. It has been proposed by Courty et al. (2017) [47] as a tool to learn how to transport data from a source domain to a target domain. Namely, given two samples X_S and X_T, the Optimal Transport Domain Adaptation (OTDA) problem is formulated as finding the transport map T from X_S to X_T. Then, a classifier learnt on $T(X_S)$ should perform well on X_T. An extension to semi-supervised learning has been proposed using a regularization

term.

Hypothesis Transfer. Hypothesis transfer learning is only applicable for semi-supervised domain adaptation. It assumes the existence of an hypothesis h_S learnt on source data which one wants to transfer to a small labeled target sample, meaning that the source data is not available for adaptation. Tommasi et al. (2010) [172] propose the LS-SVM Adaptation for the semi-supervised adaptation. First an SVM is learnt on source data and then the model for the target data is constrained to be close to the source model by constraining its parameters with an ℓ^2-norm. This idea is close to the models from multi-task learning [37].

Several models propose to refine a model learnt on source data. An early example of such method is the Adaptive SVM: Yang et al. (2007) [187] propose to add a regularization term based on target data to an SVM learnt on source data. Later, the DA-SVM [35] first learns an SVM on the source data and then iteratively adapts the margins based on the target data, in a semi-supervised fashion.

More recently, fine-tuning a neural network trained on a large related dataset on a smaller target domain has been very successful for image or text data. Namely, the idea is to freeze or constrain the first N layers of the network learnt on the source data and only fine-tune the other layers using target data. As such, this method is hybrid between hypothesis transfer and feature transfer as the feature transformation is included in the neural network. In [191], authors empirically study the transferability of image representations learnt on the ImageNet dataset. There are few theoretical studies about why this fine-tuning works and how it should be done but these techniques have been key to the recent surge of deep learning approaches. We give a more detailed overview of such methods with insights for time series in Chapter 3.

2.3.4 Summary

Following the theoretical works presented in Section 2.2, many transfer learning methods aim to minimize the distribution differences between domains, either by transferring instances, representations or hypotheses. We summarized those characteristics in Table 2.1. This overview presented general methods, but application-specific methods have already been developed and expert knowledge can largely enhance the performance of adaptation.

A question that has rarely been studied is the question of negative transfer *ie* when a domain adaptation algorithm $\mathcal{A}$ returns an hypothesis that gives a larger target risk than the hypothesis learnt on the target labeled sample. One of the reasons is that a large number of theoretical and empirical works tackle the unsupervised case where negative transfer cannot be evaluated. Some algorithms have been proposed in the semi-supervised case [184] to limit the possibility of negative transfer.

Method	Criteria	Class	Needs labeled target data?
GDM (2015, [44])	$Disc$	Instance	No
KMM (2007, [91])	MMD	Instance	No
KLIEP (2008, [168])	KL	Instance	No
Discriminative Learning (2007, [28])	Sim to $d_{\mathcal{H}}$	Instance	No
TrAdaBoost (2007, [50])	Target error	Instance	Yes
FEDA (2009, [52])	None	Feature	Yes
TCA (2011, [139])	MMD	Feature	No
JDA (2013, [113])	MMD	Feature	Yes
ARTL (2014, [114])	MMD	Feature	No
CORAL (2016, [170])	Correlation	Feature	No
GFK (2017, [76])	Subspace alignment	Feature	No
SADA (2013, [67])	Subspace alignment	Feature	No
JSADA (2015, [68])	Subspace alignment	Feature	Yes
MSDA (2011, [75])	None	Feature	No
DANN (2016, [72])	$d_{\mathcal{H}}$	Feature	No
OTDA (2017, [47])	Wasserstein	OT	No
A-SVM (2007, [187])	Margin	Hypothesis	Yes

Table 2.1: Summary of reviewed methods

2.4 Learning from Multiple Sources

In this subsection, we present an overview of methods for learning from multiple sources. Different sub-categories exist in the literature, each one of them corresponding to a different scenario:

- Multi-Task Learning: this is the case when several tasks are learnt at the same time with labeled samples, with the objective to minimize the average risk over all tasks.
- Domain Generalization: this corresponds to the ability of a model learnt on several domains to generalize well to a target domain that is not available at training time for adaptation.
- Multi-Source Domain Adaptation: this corresponds to the domain adaptation with several source domains with labeled data and a target domain with limited or no labeled data, with the objective to minimize the risk over the target domain.

This setting is different from the single-source domain adaptation as each source domain can be used to learn how to transfer. For instance, if we give ourself K source domains, then one hopes to find invariants between domains or can use a leave-one-domain-out procedure to perform cross-validation for instance.

For domain adaptation, a simple method would be to merge all the source domains to create a large single source domain. This would work under the assumption that the source domains are the same. But in many real world scenarios, some source domains share more similarities with the target domains. For instance, one would expect that the representations learnt from consumption data in cold areas of France would be more useful than the ones from warm areas of France to adapt to a target domain made of a neighbourhood from a cold area. Hence, the question of how to select or combine source domains for domain adaptation has been studied in the recent years.

We first present multi-task learning and domain generalization together as they are related. Then we present multi-source domain adaptation methods.

2.4.1 Multi-Task Learning and Domain Generalization

Framework

Multi-Task Learning. Introduced in [37], multi-task learning (MTL) aims to leverage on information from different but related tasks. The framework assumes the existence of K domains $\mathcal{D}_1, ..., \mathcal{D}_K$ with K tasks $\mathcal{T}_1, ..., \mathcal{T}_K$. In this work, we only present homogeneous multi-task learning, *ie* $\mathcal{X}_1 = ... = \mathcal{X}_K$ and $\mathcal{Y}_1 = ... = \mathcal{Y}_K$. Unlike domain adaptation, no target domain is defined and each source domain is considered as a target domain. We introduce a common representation bias $\phi : \mathcal{X} \to \mathcal{Z}$ and a class of predictors $\mathcal{H} : \mathcal{Z} \to \mathcal{Y}$. Then, noting $\mathcal{H}_\phi = \{h \circ \phi;\ h \in \mathcal{H}\}$, the objective of multi-task learning is to minimize the average risk over every source domain

$$\mathbf{R}_{\mathbf{MTL}}(\mathcal{H}_\phi) = \frac{1}{K} \sum_{k=1}^{K} \min_{h \in \mathcal{H}} \mathop{\mathbb{E}}_{(x,y)\sim P_k} [L(h \circ \phi(x), y)] \tag{2.31}$$

Domain Generalization. Domain Generalization (DG) is related to multi-task learning as it also aims to learn common knowledge from different domains. Formally, this framework assumes the existence of K domains and tasks $\mathcal{D}_1, ..., \mathcal{D}_K =$ with K tasks $\mathcal{T}_1, ..., \mathcal{T}_K$ defined over the same sets $\mathcal{X}$ and $\mathcal{Y}$. We call $\mathcal{P}$ the set of distributions over $\mathcal{X} \times \mathcal{Y}$. Then domain generalization assumes that every joint distribution $P_k(X_k, Y_k)$ is drawn from a distribution $\mu \in \mathcal{P}$. Similarly to MTL, we introduce a common representation bias $\phi : \mathcal{X} \to \mathcal{Z}$ and a class of predictors $\mathcal{H} : \mathcal{Z} \to \mathcal{Y}$. Then, noting $\mathcal{H}_\phi = \{h \circ \phi;\ h \in \mathcal{H}\}$, the goal of domain generalization is to minimize the risk over any possibly unseen domain drawn from μ

$$\mathbf{R}_\mu(\mathcal{H}_\phi) = \mathbb{E}_{P\sim\mu} \left[\min_{h \in \mathcal{H}} \mathop{\mathbb{E}}_{(x,y)\sim P} [L(h \circ \phi(x), y)] \right] \tag{2.32}$$

One can see that the risk of multi-task learning is a biased estimator of the risk of domain generalization. In our definitions, we included the common representation bias ϕ for more clarity but in more general definition, it is absent from the hypothesis class and only implicit. Hence, solving the multi-task learning or domain generalization problem comes back to learning the common bias between every source domains.

Domain generalization bounds

A first question is: why do multi-task learning and domain generalization work? In [22], Baxter proved a formal generalization bound between $\mathbf{R}_{\mathbf{MTL}}$ and $\mathbf{R}_\mu$ using covering numbers and get a bound of the form $\mathbf{R}_\mu \leq \mathbf{R}_{\mathbf{MTL}} + \epsilon$ where ϵ depends on the number of domains K, the number of labeled samples n and the complexity of the class

$\mathcal{H}_\phi$. From the main theorem of this work, one can deduce that if the number of domains K and the number of labeled samples n per domain are sufficient, then any learner that learns the representation bias ϕ can bound the generalization error. This also means that it is possible to have an estimate of the ability of such a bias to generalize to an unseen domain using the K available domains. Moreover, they also show that the number of training examples per domain required to get a tight generalization bounds decrease as the number of domains grows, which confirms the intuition that leveraging from different domains helps to learn a good hypothesis. Finally, they show how it applies to feature maps learnt with neural networks.

This work confirms many of the intuitions proposed in the seminal work of Caruana [37]. Namely, those intuitions can be summarized as the fact that the different tasks share the same important features. Then multi-task learning helps each domain to learn those features with auxiliary data coming from the other domains. This ability to transfer features via sparse coding has also been proven in [125] [115].

Methods for MTL and DG

Block-Regularization. The main line of work of multi-task learning is to enforce a regularization between the different hypotheses learnt on each domain. In general, it takes the form of block regularization, as in [63] [10]. Authors propose a sparsity constraint between tasks with linear classifiers in the form of an ℓ^1/ℓ^2-norm. Namely, given K samples $\mathcal{S}_k = \{(x_k^{(1)}, y_k^{(1)}), ..., (x_k^{(n)}, y_k^{(n)})\}$ with $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$, they propose the following objective

$$\min_{W \in \mathbb{R}^{K \times d}} \sum_{k=1}^{K} \sum_{i=1}^{n} L(y_k^{(i)}, w_k^T x_k^{(i)}) + \lambda \|W\|_{2,1}^2 \tag{2.33}$$

where $\|W\|_{2,1}^2 = \sum_{i=1}^{d} \|w_i\|_2$ enforces a sparsity in the columns, *ie* only a few features are selected for every domain. This regularization has been very popular for multi-task learning and has been extended with block-sparse dictionary learning [134] and other sparsity constraint such as hierarchical constraints [12] or mixed sparsity constraints [97].

Low-Rank assumption. Opposed to the sparsity constraint, some methods propose to use a low-rank assumption, *ie* the domains share parameters lying on a same subspace. Those methods can broadly be summarized with the framework of [8] for each task as $w_k = u_k + Av_k$ where $u_k \in \mathbb{R}^d$ is a domain-specific parameter and $A \in \mathbb{R}^{d' \times d}$ with $d' << d$ is a shared subspace between every domain. This low-rank assumption has later been made using a trace-norm regularization [145].

Parameter sharing. Using neural networks, multi-task learning takes the form of parameter sharing. Namely, the common bias learner ϕ is made of the first layers of a neural network, while final predictors $h_1, ..., h_k$ are different

for every task, as proposed in [22]. From there, other methods have proposed different ways of mixing the common bias, for instance via a regularization between the common layers [60].

Learning Task Relationships. Some methods make the assumption that all the tasks do not share the same relationships with each other, and that they are organized in clusters. Two main lines of work tackle this scenario: a first line of work first clusters the tasks based on a given criterion and then uses multi-task learning on these different clusters. Another line of work performs both clustering and multi-task learning at the same time. Many of these methods perform a clustering on the parameters of each task such as in [96]. Other methods use hierarchical sparse coding to learn a hierarchical clustering structure for multi-task learning [200] [82].

Causality. Another related topic is causality [142]. While the formalism of causality is slightly different from domain generalization, it shares some similarities. In general, searching for causality relationships between different variables comes back to finding invariant relationships when changing environments. This link has been studied in [154] where authors try to find the features that are invariant to a change of domain. The derived algorithm is not scalable and the hypothesis for it to work make it hard to use in practice but this is a promising line of work.

Summary In this section, we presented multi-task learning and Domain Generalization, which are related to Domain Adaptation. We saw that it is possible to learn from multiple domains to generalize to a new domain without adaptation. Different models were proposed to learn what is the common knowledge shared in different tasks. Now, similarly to what single-source domain adaptation tries to tackle compared to out of sample generalization, multi-source domain adaptation raises the issue of adaptation to a new domain when a sample from this new domain is available.

2.4.2 Multi-source domain adaptation

Framework Domain Adaptation with Multiple sources (MSDA) assumes that K source domains $\mathcal{D}_1, ..., \mathcal{D}_K$ are available with a target domain $\mathcal{D}_T$ with associated tasks $\mathcal{T}_1, ..., \mathcal{T}_K, \mathcal{T}_T$. Once again, we only present homogeneous transfer learning, *ie* $\mathcal{X} = \mathcal{X}_1 = ... = \mathcal{X}_K = \mathcal{X}_T$ and $\mathcal{Y} = \mathcal{Y}_1 = ... = \mathcal{Y}_K = \mathcal{Y}_T$. For each source domain, a labeled sample $\mathcal{S}_k = \{(x_k^{(1)}, y_k^{(1)}), ..., (x_k^{(n)}, y_k^{(n)}\}$ is available. For unsupervised MSDA, the target domain does not have a labeled sample, whereas for semi-supervised MSDA, the target domain has a small labeled sample and possibly a large unlabeled sample.

Domain Selection A first class of methods assume that not every domain is useful for the adaptation and aims to select the best domains. Indeed, transferring from unrelated domains can lead to negative transfer [155]. In [48] authors propose a theory to select the K^* best sources based on a convergence bound. Then they use the average

of the classifiers learnt on each selected source as the final classifier. Recently, authors of [20] proposed a source selection method based on distance between domains. We will see that those methods can be seen as a special case of the combining methods with binary weights.

Combining domains Another class of methods assumes that the domain distributions can be combined with a linear combination. In [25], authors introduce the α-weighted source domain as $\mathcal{D}_\alpha = \{\mathcal{X}, p_\alpha\}$ such that $p_\alpha = \sum_{k=1}^K \alpha_k p_k$ with $\sum_{k=1}^K \alpha_k = 1$ and $\alpha_k \geq 0$. Then it is possible to prove a bound on the target risk of a minimizer $\widehat{h_\alpha} \in \mathcal{H}$ of the α-weighted source risk $\widehat{\mathbf{R}_\alpha}(h) = \sum_{k=1}^K \alpha_k \widehat{\mathbf{R}_\mathbf{k}}(h)$ similar to the bound 2.16:

Theorem 9. *Using the previous notations, if* $h_T^* = \min_{h\in\mathcal{H}} \mathbf{R_T}(h)$*, then with probability* $1-\delta$ *over the sampling* K *samples of size* m $\{\mathcal{S}_1, ..., \mathcal{S}_K\}$*, we have*

$$\mathbf{R_T}(\widehat{h_\alpha}) \leq \mathbf{R_T}(h_T^*) + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T) + 2\gamma_\alpha + 4\sqrt{\left(\sum_{k=1}^K \alpha_k^2\right)\left(\frac{VC_\mathcal{H}\log(2m) - \log\delta}{2m}\right)} \tag{2.34}$$

where $\gamma_\alpha = \min_{h\in\mathcal{H}}\left[\mathbf{R_T}(h) + \sum_{k=1}^K \alpha_k \mathbf{R_k}(h)\right]$.

From this theorem, we can conclude that learning weights that minimize the divergence between domains $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T)$ helps for domain adaptation. After the weights are learnt, one only needs to learn one hypothesis on the re-weighted domain. Such an approach was used in [39], where the weights α were learnt using a manifold regularization. One work of this thesis [152] develops a similar framework.

Combining models Finally, a last class of methods combines models with weights. For instance, in [122] authors show that if the target distribution can be written as a mixture of every source distribution (*ie* $P_T = \sum_{k=1}^K \alpha_k P_k$), then the hypothesis $\tilde{h_\alpha} : x \to \sum_{k=1}^K \frac{\alpha_k P_k(x)}{\sum_{l=1}^K \alpha_l P_l(x)} h_k(x)$ is such that $\mathbf{R_T}(\tilde{h_\alpha}) \leq \max_k \mathbf{R_T}(h_k)$. While this bound is quite loose, it means that with this re-weighting, one cannot learn a worse hypothesis than by selecting one of the h_k, which does not hold for the linear combination $\sum_{k=1}^K \alpha_k h_k$.

In [88], authors extended this work with the Rényi divergence and derived an algorithm to find these weights when they are unknown. A weighting scheme of predictors learnt on each domain was also proposed by using weights that represent the likelihood of each source sample to come from the target distribution, as in [171]. Another work [196] used the bound 2.16 for multiple source adaptation by iteratively adapting a neural network to the domains that is the furthest using adversarial learning. While counter-intuitive, this method works when there is no source domain that is far from the target domain ; otherwise, it is subject to negative transfer.

Summary The most common way to combine sources to adapt to a target is to linearly combine them. This combination can be done using the distributions or predictors learnt on each hypothesis. As was shown in the bound of Theorem 9, to avoid negative transfer a low weight must be given to domains far from the source. This is

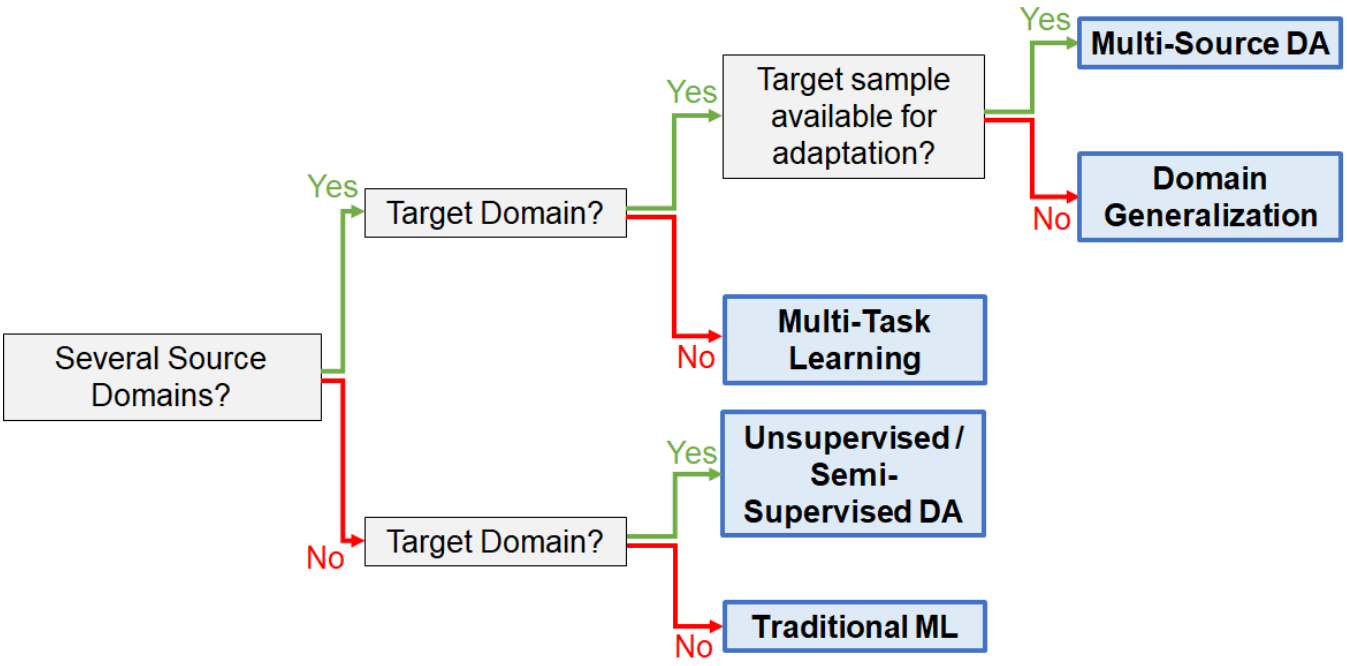


Figure 2.5: Scenarios of Transfer Learning with or without multiple sources

what source selection aims to do by giving a 0 weight to sources that would bring negative transfer. Finally, one can also directly use the methods from multi-task learning and domain generalization without further adaptation, which would work when every domain shares the same relationship with the task.

2.5 Conclusion

Figure 2.5 presents a synthetic view of the methods reviewed in this Chapter. Transfer learning is a very large field and some methods are missing from the review, but we focused the most relevant subfields for the work of this thesis. We observe that there are many ways to adapt, both from a theoretical point of view, but also depending on what kind of data is available.

In the computer vision and natural language processing (NLP) communities, the most popular transfer learning methods are based on fine-tuning a publicly available pre-trained neural network to the small target dataset [191, 188]. While empirical results are extremely good, new research on the theoretical understanding of this method would greatly contribute to the field.

Clearly, a large pre-trained neural network is not available for every applications and more traditional methods are still popular. For instance, [129] propose a tree-based transfer learning model for fall detection by transferring from actors to elderly people or Ma et al. (2015) [117] use TrAdaBoost [50] to estimate dust aerosol based on satellite images by transferring from one satellite to another.

In the next Chapter, we review time series representations. Indeed, transfer learning is largely dependent on the

representation or the data. As one wants to match source and target distributions, it is necessary to define good source and target features, which is not straightforward for time series.

Chapter 3

Deep Time Series Representations for Non-Intrusive Load Monitoring

Contents

Studying household consumption device by device is a long-standing issue in machine learning applications for energy. It could potentially enhance the predictability of electricity consumption, which would lead to a reduction in electricity waste to help mitigate climate change and increase the efficiency of the energy industry. Moreover, household electricity consumption is a key factor for many applications of EDF as predicting the consumption is necessary to adapt the production. With the deployment of smart meters for electricity consumption in France, one could hope to extend the analysis made of a client's consumption, for instance for customized contract to give them incentive to smooth their consumption.

As detailed below, collecting data on each device in each household is costly. It means that many datasets have been collected over separate experiments and therefore they are not homogeneous. The absence of large and clean datasets makes hard to learn very general features, as there is a huge variety of households, devices and consumer behaviours. Transferability from a household to another is not guaranteed and we present in this thesis solutions to improve transferability between households.

Time series analysis is firstly reviewed, with a special interest on deep time series representations and their transferability. The issue of Non Intrusive Load Monitoring is then presented along with details explaining why Transfer Learning is required. Finally we present different datasets, both collected at EDF during this thesis and public datasets.

This work first focuses on the sub-problem of appliance recognition: from the consumption data of one appliance, is it possible to recognize the nature of the appliance? We observe that for this problem, there is already a performance drop if the algorithm is learnt on some appliance models then applied to recognize new unseen models of the same appliances.

To mitigate this issue, we propose a novel deep-learning based method with a simple time series normalization solution that allows to learn more invariant features for appliance recognition. Extensive experiments are conducted on different datasets and show the benefits of our approach.

This chapter is partly based on two works published in two conferences:

- G. Richard, G. Hébrail, M. Mougeot and N. Vayatis. "DenseNets for Time Series Classification: towards automation of time series pre-processing with CNNs." MiLeTS19SIGKDD 25th ACM SIGKDD International

Conference on Knowledge Discovery & Data Mining (KDD'2019)

- G. Richard, B. Grossin, G. Germaine, G. Hébrail and A. de Moliner. "Autoencoder-based time series clustering with energy applications." Conférence sur l'Apprentissage Automatique 2018 (CAp 2018).

3.1 Background on Time Series Representations

A time series is an ordered sequence of values of a variable observed at given time intervals. Time Series Analysis is a long standing field as time series arise in many real world data: weather, finance, audio processing, energy, ... Many methods have been proposed for various tasks on time series data, such as classification, forecasting, clustering, ... The time dependency makes the analysis of this data a real challenge. Moreover time series generally are high dimensional and subject to noise.

This time-dependence and high dimensionality requires to find new representations to use instead of raw time series. After presenting the general definitions and notations, we review some classical time series representations methods for machine learning applications such as classification and regression.

3.1.1 Framework

A time series is generally defined by a sequence of ordered points taken at successive timestamps. Namely it is a sequence $\{X_{t_1}, ..., X_{t_n}\}$ where t_1, ..., t_n are the sampling times and X is a vector. When the sampling times are regular, *ie* $t_2 - t_1 = t_3 - t_2 = ... = t_n - t_{n-1}$, one can define a multi-variate time series as an ordered sequence of vectors $X = [X_1, ..., X_T] \in \mathbb{R}^{d \times T}$ where d is the dimensionality of the time series and T its length. For instance, in industrial system monitoring, d is the number of sensors. When $d = 1$, we speak of univariate time series.

We consider the problem of time series binary classification but these feature engineering methods can be extended to regression or clustering. This means, that in general, we have a dataset $\mathbf{X} = \{(X^{(1)}, y^{(1)}), ..., (X^{(n)}, y^{(n)})\}$ where $X^{(i)} \in \mathbb{R}^{d \times T}$ is a time series and $y^{(i)} \in \{0, 1\}$. Moreover, we assume the different $X^{(i)}$ to be i.i.d.

The time-dependence of time series means that simply treating each point as a separate feature and apply multivariate data analysis would be sub-efficient. Hence, different feature transformations have been proposed to include the time-dependence.

3.1.2 Overview of Univariate Time Series Representations

In this section, we only consider uni-variate time series as we focus on applications on (uni-variate) electricity consumption in Chapter 3 and Chapter 4. Some of the methods presented below can be extended to multivariate time series. In the following we denote a time series dataset $\mathbf{X} = \{(X^{(1)}, y^{(1)}), ..., (X^{(n)}, y^{(n)})\}$ where $X^{(i)} \in \mathbb{R}^T$.

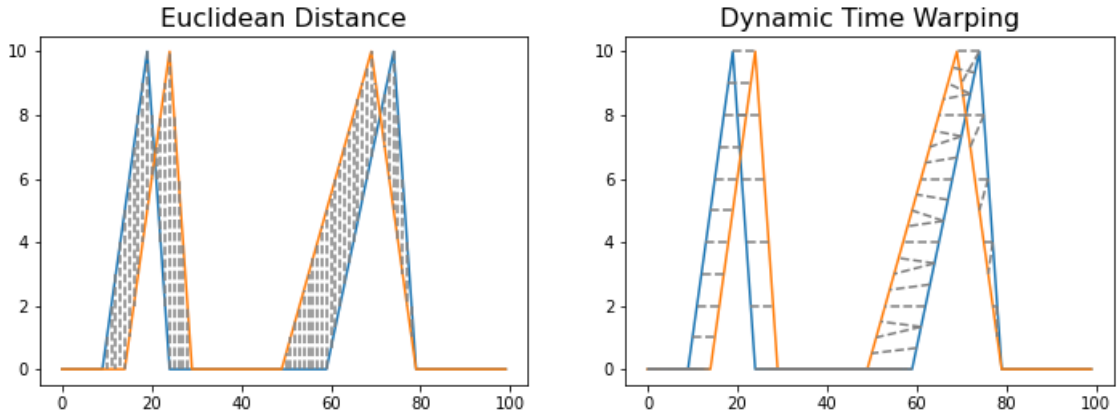


Figure 3.1: Euclidean Distance vs Dynamic Time Warping

Time Series Distance

Even if distances are not feature representations, they are a popular tool for time series classification, and stress why defining specific tools for time series is important [58]. A first possible distance to compare two time series is the traditional Euclidean distance:

$$d(X^{(i)}, X^{(j)})^2 = \sum_{t=1}^{T} (X_t^{(i)} - X_t^{(j)})^2 \tag{3.1}$$

The issue with this distance is that it is not robust to time shift in the data: for instance, in Figure 3.1, a small temporal is present between the plotted time series. The Euclidean Distance is high as it is calculated time stamp by time stamp. Hence, Dynamic Time Warping [26] was proposed to allow some temporal shifts in the data. Namely, to compute the distance between two time series $X^{(i)}$ and $X^{(j)}$, a path $W = \{w_1, ..., w_K\}$ is defined such that $w_1 = (1, 1)$, $w_K = (T, T)$ and for $w_k = (t, s)$ and $w_{k+1} = (t', s')$, we have $t \leq t' \leq t + 1$ and $s \leq s' \leq s + 1$. If we define $\mathcal{W}$ the set of paths that respect these conditions, the Dynamic Time Warping distance is defined as:

$$d_{DTW}(X^{(i)}, X^{(j)}) = \min_{W \in \mathcal{W}} \sum_{k=1}^{K} (X_{t_{w_k}}^{(i)} - X_{s_{w_k}}^{(j)})^2 \tag{3.2}$$

This is solved by dynamic programming and allows for shifts as showed in Figure 3.1. Dynamic Time Warping is still considered a very strong baseline for uni-variate time series classification and pattern retrieval [58]. The main issue of Dynamic Time Warping is its computation time although numerous methods propose tricks to accelerate its computation [132].

Statistical features

A very common way to extract features from time series is to use its statistical features. Feature mapping has been extensively studied for time series analysis. Nanopoulos et al [133] propose a method where different statistical features from time series are extracted for classification. These features are similar to these used in exploratory

data analysis such as mean, variance, skewness, kurtosis, minimum, maximum, percentiles, ... at different orders (features extracted from $S_{t+D} - S_t$ where D is a user-defined value).

Classically, the strong assumption made for such transformations is that the time series respect stationarity, which means the distribution of successive data points does not change over time. Wide Sense Stationarity is more widely used where only the first and second order moment of the distribution must stay constant.

These conditions are rarely respected in real-world applications but this kind of feature transformation is still a strong baseline. A way to circumvent this issue is proposed in [23] where authors propose a bag-of-features model by extracting features from subsequences with a sliding window. While the whole time series is not stationary, one could expect smaller subsequences to be.

This class of methods is very useful as an exploratory tool. Indeed, it is generally easy to use and can be used as a first baseline. Moreover for many classifiers, the method is interpretable as feature importance can be analyzed. Finally it is possible to create features for specific problems with expert knowledge, enhancing the performance of the method.

Dictionaries

Fixed Dictionary For many signal analysis application, designing a basis on which to decompose the signal allows to find relevant patterns. For simplicity we only present some basis function often used as a pre-processing step for machine learning applications. For our applications, we summarize these methods as follows:

- Choose a basis of functions $D \in \mathbb{R}^{T \times T}$
- Project X on D *ie* $X = \bar{X}D$ with $\bar{X} \in \mathbb{R}^{T}$
- Use coefficients $\bar{X}$ as features.

In general, links can be made with functional analysis [147] where time series are treated as a function. Fourier basis [31] for frequency analysis and Wavelet basis [119] for scale and frequency analysis are very popular methods for time series analysis. For electricity consumption, spline basis has been used successfully [186]. Coefficients obtained from such decomposition are applied successfully for time series classification in [161] [160] where the spectrogram is used to create a symbolic representation of the time series.

Dictionary Learning For the method presented before, the functional basis is pre-defined, based on an insight on the underlying nature of the signal. It is also possible to learn it using dictionary learning [3]. In general, finding a basis to decompose a signal is an underdetermined problem so one has to guide the learning to find appropriate solutions.

We consider a dataset $\mathbf{X} \in \mathbb{R}^{n \times T}$ of n time series of length T. In general, the basis $D \in \mathbb{R}^{f \times T}$ to learn can be overcomplete ($f >> T$) or undercomplete ($f << T$). When the base is overcomplete, a sparsity constraint is imposed on the coefficients, for instance

$$\tilde{D} = \min_{D} \|X - AD\|_2 + \lambda \sum_{i=1}^{n} \|A_i\|_1 \tag{3.3}$$

Efficient solutions have been proposed to solve this problem [118]. Undercomplete dictionary is also often called matrix factorization [109]. In general, overcomplete dictionaries are useful for signals that can be decomposed by a small number of a large collection of patterns and undercomplete dictionaries for signals that are a combination of a small number of common signals [59].

Auto-regressive models

Other typical features for time series data are derivated from autoregressive models: one of the most popular is the AutoRegressive Moving Average (ARMA) model. It states that the time series follows the following model :

$$S_t = C + \epsilon_t + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{i=1}^{p} \theta_i \epsilon_{t-i} \tag{3.4}$$

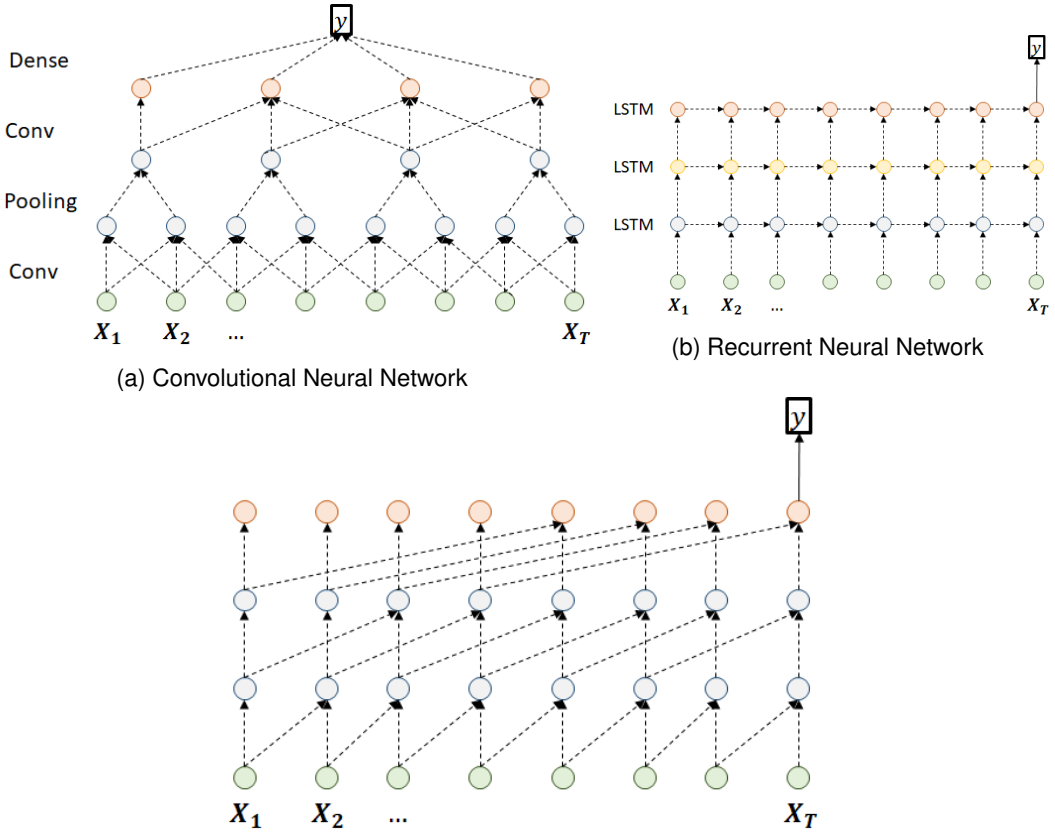
where p and q are hyperparameters, $\epsilon_.$ are white noises and $\phi_.$ and $\theta_.$ are the parameters of the models that have to be fit on the time series. These two set of parameters are then used as features, such as in [13].

3.2 Transferability of Deep Time Series Representations

3.2.1 Deep Time Series Representations

Deep Learning has been very popular in recent times for unstructured data such as images or text, finding applications in object recognition [103], text translation [177] or text generation [162]. Hence, some architectures are designed specifically for time series, reviewed in [65]. The two main architectures are convolutional neural networks (CNN) and recurrent neural networks (RNN). Deep neural networks simultaneously learn representations and a classifier.

Broadly speaking, RNNs are networks learning from sequences, where each neuron feeds the next one with a memory parameter so that prediction at time t depends of the whole sequence. RNNs are extremely popular for NLP applications where state-of-the art methods are based on recurrent neural networks. For time series, they are rarely applied for several reasons. Firstly, typical time series are very long, and methods such as Gated Reccurent Unit (GRU) [42] or Long Short Term Memory (LSTM) [87] struggle to handle long sequences. Secondly, time series



(a) Convolutional Neural Network

(b) Recurrent Neural Network

(c) Temporal Convolutional Network

Figure 3.2: Different architectures for time series classification

are often subject to noise where text data has in general a better structure. Some RNN are proposed for time series [71] but appear to be underperforming in general [65].

CNNs are similar to typical neural networks but use convolutional layers, meaning that these layers keep some temporal information from the input. This kind of layers can help to identify discriminative patterns in time series, as for the popular shapelet method [190]. Links have also been established between convolutional neural networks and wavelet analysis [120]. CNNs are very popular in computer vision, with popular architectures such as AlexNet [103] or VGG16 [165]. They are generally made of convolutional layers and fully connected layers defined as follows: a fully connected layer (also called dense layers) takes an input $z \in \mathbb{R}^n$ and transforms it into an output $y \in \mathbb{R}^l$ through an activation function σ :

$$y = \sigma(Wz + b) \tag{3.5}$$

where $W \in \mathbb{R}^{l \times n}$ and $b \in \mathbb{R}^l$ are respectively the weight matrix and the bias of the layer to be learned. Similarly a convolutional layer is defined by a transformation :

$$y_i = \sigma(\sum_{j=1}^{r} W_j z_{1+(i-1)s+j} + b_j) \tag{3.6}$$

where $W \in \mathbb{R}^r$ and $b \in \mathbb{R}^r$ are respectively the kernel matrix and the bias of the layer to be learned with kernel size r and stride s. For a classification task, convolutional layers are stacked before using fully connected layers for prediction. The parameters of the network are then trained in an end-to-end fashion by minimizing a loss L on training samples and using feed-forward backpropagation. More details about neural networks can be found in Appendix A.

The temporal structure of a time series is taken into account in a CNN as the convolutions keep temporal information as shown on Figure 3.2. From classical CNNs, many works tried to derive specific architecture for time series. Fully Convolutional Networks [183], Multi-scale Convolutional Neural Network [49], LeNet [106] are all proposed based on traditional neural networks architectures. Current state-of-the-art methods[1] on the popular UCR dataset [51] for time series classification are ResNet [183] and DenseNets [151] (see Section 3.4) which are both methods including skip connections.

In order to better take into account the temporal nature of the data, temporal convolutional network (TCN) [14] have been proposed as an extension of WaveNet [137]. They extend the notion of CNN by including causal dilated convolution. Using causal convolutions means that, at layer k, the neuron at position t only receives information from neurons at position $j < t$ at layer $k-1$:

$$y_i = \sum_{j=1}^{r} W_j x_{i-jd} \tag{3.7}$$

where r is the kernel size and d the dilation rate. In Figure 3.2, one can observe that TCN is hybrid between CNN and RNNs.

Finally, CNN can also be used for feature extraction in an unsupervised way with autoencoders illustrated in Figure 3.3. Autoencoders are a class of unsupervised methods used to learn a new data coding. Deep autoencoders has led to numerous works in recent years and gave promising results for time series analysis [105]. In Appendix B, we show an application of autoencoders for consumption clustering.

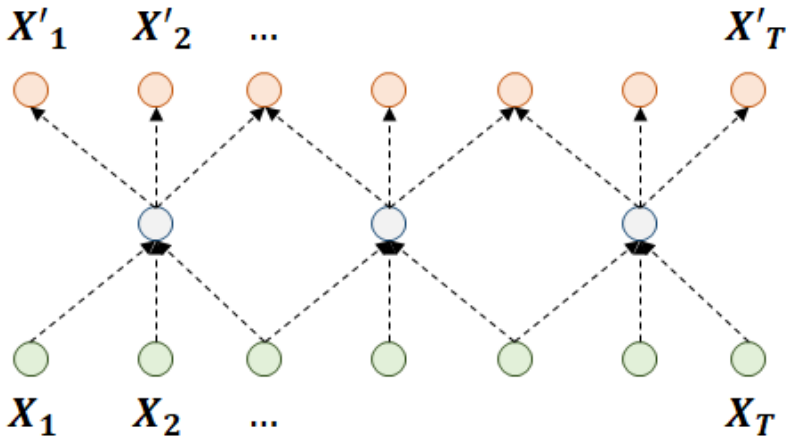


Figure 3.3: Convolutional Auto-Encoder

[1] Better results can be obtained using ensembling such as [112] [66]

3.2.2 On Transferability of Deep Time Series Representations

A reason for the popularity of deep learning in image and text is that the transferability or generalization ability of representations learnt on large datasets, such as ImageNet [56] for computer vision or Word2Vec embeddings learnt on Twitter feeds [144] or a mixture of large corpus [57]. With a small number of labeled images, one can use an architecture such as AlexNet to fine-tune the network as mentioned in Section 2.3.3. This idea is the key to many sucessful applications of deep learning techniques today, as learning such architectures require a large amount of data.

A natural question is: can one find such large architecture and representations for time series? The question has been studied in recent works. Fawaz et al. (2018) [64] use the UCR archive to empirically study the transferability of pre-trained neural networks. They observe that in general, transferring from one dataset to another lead to negative transfer but if the source dataset is well chosen, it can improve the classification accuracy.

One must note that the UCR archive is made of datasets of time series of various length and coming from very different fields: some are shapes transformed to time series, others come from medical applications (EEG/ECG), others are electricity consumption, ... Moreover, the size of the datasets is generally small, between 20 and 8926 time series with only 11 out of 128 datasets made of more than 1000 time series. As a comparison, the ImageNet dataset is made of more than 14 million images. UCR archive is currently the largest database for time series classification and a very useful tool to experiment a new method, but is not suited to assess the transferability of pre-trained neural networks. One could even argue that with datasets made of 16 time series, most neural network architectures would be prone to overfitting.

In summary, the results on transferability of pre-trained neural networks obtained in computer vision are hard to assess for time series, mainly for the lack of a ImageNet-like time series database. Intuitively, we still expect that given a large dataset of time series coming from a specific application (consumption data in France for instance), one could hope to learn a deep CNN on this dataset and use this pre-trained CNN on a similar dataset (consumption data in Belgium for instance). For instance, Song et al. (2019) [166] tried to collect such a dataset for gait movement data. With limited data, there is a need for more advanced transfer learning methods than fine-tuning from a pre-trained neural network.

3.3 Transfer Learning in Non Intrusive Load Monitoring

3.3.1 General presentation

Non Intrusive Load Monitoring (NILM) is the field of analysis of the consumption of electric appliances in a household. It is introduced by Hart (1989) [85] as the process to analyze changes in the voltage of different appliances with the goal of knowing which appliances are turned on or off.

Generally speaking, the goal of NILM is to the disaggregation of the load curve of a household. Formally, if we call $X \in \mathbb{R}^T$ the total consumption of a home over a period T, the goal of NILM is to predict the consumption at each timestamp of each appliance in the household $Y = [X_1, ..., X_A]$ where A is the number of appliances (generally unknown). One can note that the whole consumption is equal to the sum of the sub-consumptions of each appliance, hence NILM is similar to a source separation problem.

As such this is a very hard problem for several reasons. Firstly, to solve it perfectly, one must know every appliance in a home. It is almost impossible as some appliances such as phone charges are not plugged all the time. In general, one only considers some high-consuming appliances such as washing machine, dishwasher, electric heater, water heater, ... Secondly, the problem of source separation is known to be already hard when the number of measured signals is the same as the number of sources. Here, we only have one output signal.

As presented in Figure 3.4, many interesting sub-problems exist for Non Intrusive Load Monitoring. The most basic problem is to know if an electric appliance can be recognized from its own consumption. This is known as appliance signature recognition, where a signature is a single cycle of consumption of an appliance. A more complicated problem is to detect if an appliance is turned on or off at a given timestamp. This is often formulated at a sequence classification by considering a window around the timestamp. More complicated is to predict the consumption of an appliance for a given time window. For instance one could consider the problem of predicting the daily consumption of a fridge from the whole house consumption over a day. These sub-problems are simpler to solve than the original disaggregation and are sufficient for many applications. Moreover, the complete disaggregation of the consumption of a household raises the issue of privacy. Solving simpler problems might mitigate this privacy issue.

Data collected for NILM can take two forms that lead to very different methods. The first class of data collected is made of only power measurements at a sampling of a second or more. This can be done using a smart meter to monitor house consumption and specific plugs for each device. The other class of data are high frequency (higher than 1000Hz) measurements of power and/or voltage. This allows to work on both active and reactive power and to detect appliance activations using their complex impedance.

In this work, we focus only on the first class of data, as it is the data collected in the experiments led by EDF. Moreover, most houses are not equipped with high frequency sensors, so the applicability of such methods is limited. Such methods seem more suited for commercial and industrial buildings, where the energy saved can be worth the investment.

Appliances are often categorized in four categories:

- On/Off appliances: these appliances are manually turned on or off and have two states, such as lamps, computers, TVs, ...
- Finite State appliances: these appliances have multiple operating states, such as washing machine, dish-

washer, ...

- Permanent and cyclical appliances: these appliances have a permanent or cyclical unique operating state, such as fridge, smoke detectors, ...
- Continuously Variable Appliances: these appliances do not have fixed operating states and are very hard to process, such as electrical heater.

Many methods treat each appliance separately [141] and some appliances are harder to analyze. For instance, EDF experts agree that continuously variable appliances such as electrical heaters are very hard to disaggregate whereas regular appliances such as the fridge should be easier to analyze.

NILM opens several sources of potential shifts between different houses. The first source of shift is the differences between appliances both in their cycles and in their power load. For instance, appliance consumption is higher in the US than in France. Moreover, some devices are very different: for instance, in the US, water is already hot when arriving in the washing machine, whereas in France, a first cycle is required to heat the water. Secondly, there are differences in behaviour. The usage of a washing machine is very different for single people from large families. This means that when learning a disaggregation algorithm on one house, it may be under-performing on another house.

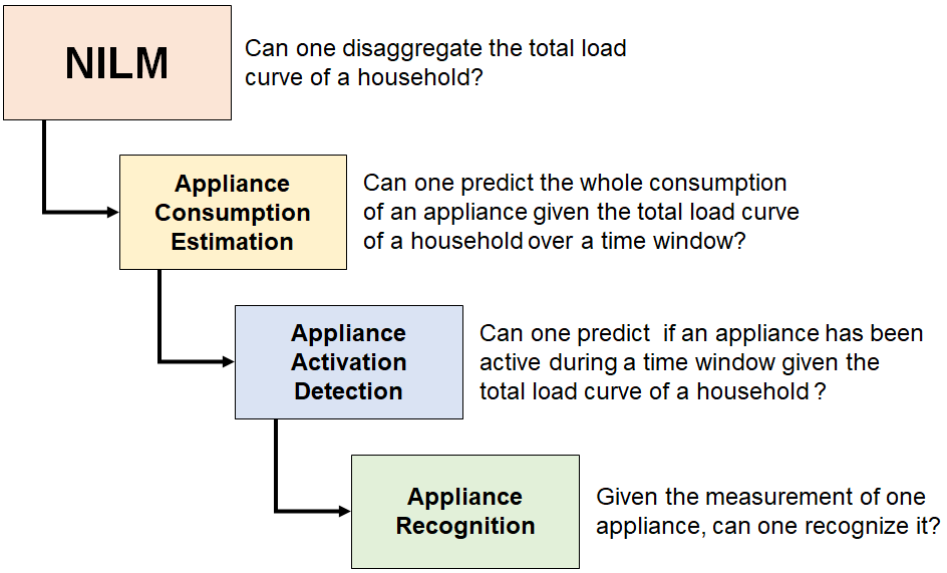


Figure 3.4: Sub-problems of Non Intrusive Load Monitoring (NILM)

3.3.2 Review of methods

The seminal work of Hart (1989) [85] proposed combinatorial optimization to solve the original problem but requires power and voltage data. Later, Batra et al. (2013) [21] proposed to use combinatorial optimization for single power

measurements later by comparing the signal to a fixed database of appliance signatures and using methods similar to dictionary learning. This approach can work when the number of appliances is small but is not robust to continuously variable appliances and does not scale well with the number of appliances. Hence, when electric heater is present, it is very hard to detect other motifs. Dictionary Learning is also proposed for load disaggregation but suffers even more from these issues [102].

Parson et al. (2012) [141] opened a new line of work based on Factorial Hidden Markov Models. The basic idea behind this line of work is to model the state of an appliance using a HMM. For instance, ON/OFF appliances are modeled by two states and finite state appliances by their number of states, where each state is described by a gaussian distribution. This class of models is later extended using more complex models [198]. These methods are currently a strong baseline for disaggregation, although they require expert knowledge on the *a priori* power distributions of the appliances. Such knowledge is possible when every device is known but there is a huge variability in devices depending on brands, countries, ... They are also weak to continuously variable appliances, but perform extremely well for cyclical appliances such as the fridge for instance.

Deep Learning is the third main branch of NILM methods. Originally proposed by Kelly and Knottenbelt (2015) [100], several models are proposed. The setup uses data from the publicly available REDD experiment (power measurement sampled at 1 minute): they select 5 appliances and consider the problem of predicting the load curve of each appliance based on the total consumption during a week. The first model learns a *seq2seq* architecture traditionnally used for denoising based on an RNN and a convolutional autoencoder. It is similar to an encoder where the input is the total load curve and the output the load curve of one appliance. The second model simply predicts the start and the end of an activation with its average value. These models outperform the previously mentioned models on the REDD dataset and are currently considered as state-of-the-art. These promising results opened several works, both academic [193] and industrial at EDF for instance, including the ones of this thesis. The main disadvantage of this class of method is the requirement of a large labeled dataset.

Finally, an emerging class of method is based on Graph Signal Processing [159]. In Graph Signal Processing, each appliance is considered as a node of a graph and its consumption is a time series. In [195], authors propose an unsupervised disaggregation method based on graph signal processing, which works well if the appliances have mean power very different from each other. This line of work is not the best performing but is promising.

3.3.3 Datasets

Electric Data (EDF)

The main dataset of interest for our work is collected by EDF firstly in 2016 and in 2018 and 2019 (during this thesis). In total, 27 households are monitored during several months, both at the whole consumption level and the appliance level. It is done using only the house electricity meter: a sensor monitors the whole consumption every second and

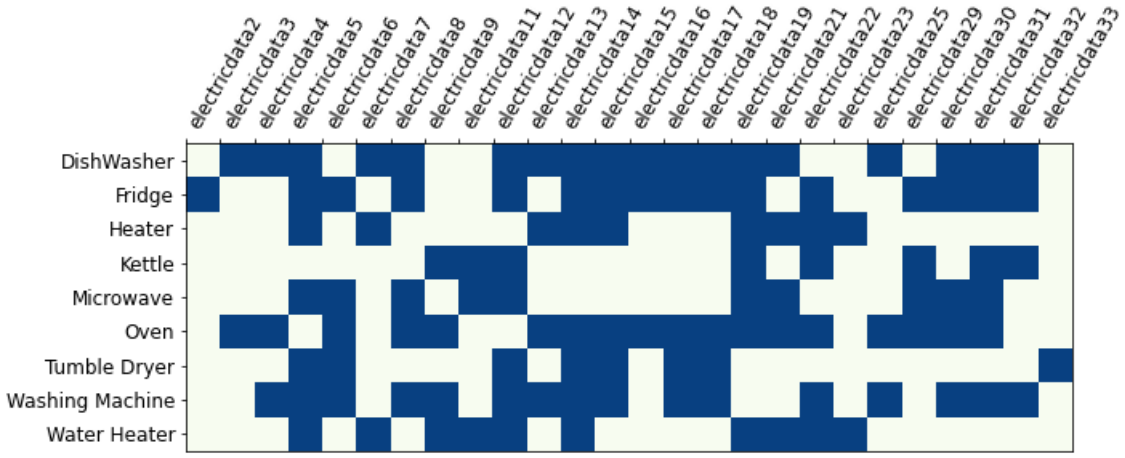


Figure 3.5: Examples of monitored appliances for each house: a blue square means the appliances was monitored. A white square means it was not monitored or was mixed with another appliance.

7 sensors are plugged to the circuit breaker corresponding to some large appliances:

- Dishwasher
- Fridge
- Tumble dryer
- Washing Machine
- Oven
- Water heater
- Electrical Heater.

Each house has different appliances and some may be absent from some houses. The details about the composition of each house is available in Figure 3.5. Unfortunately, the collected data does not allow to analyse smaller appliances. Indeed, they are too often mixed with larger appliances as two appliances can be connected to the same circuit breaker. For this reason, in our experiments, we are limited to few devices. On Figure 3.6, the consumption over one day of different appliances is represented. One can observe that heating appliances are responsible for a large part of the consumption and they are very noisy.

Public datasets

Public datasets are also used in our experiments. A large number of datasets exist but we decide to keep the one that are closer to our experiments in terms of measurements: REFIT and TraceBase. REFIT is a dataset made of cleaned consumption data in Watts for 20 households at aggregate and appliance level, sampled at a various

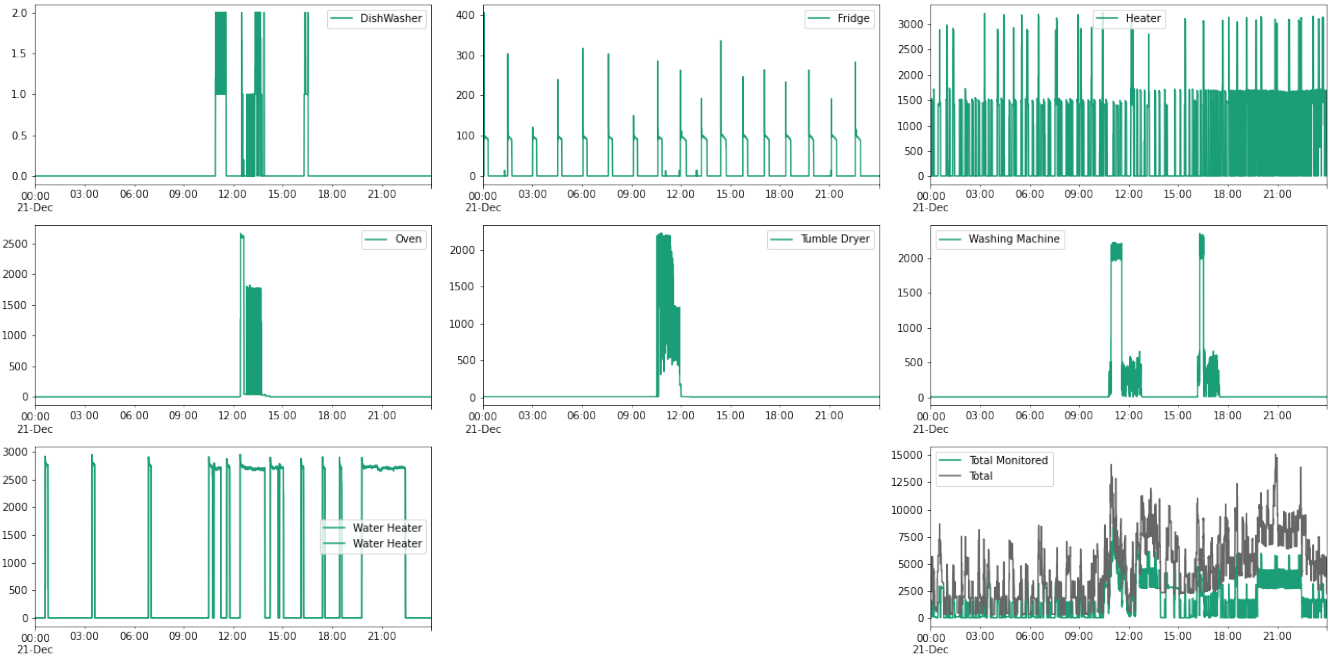


Figure 3.6: Examples of monitored consumption for one day in the Electric DataBase (x-axis is in hours and y-axis is in W)

interval (around $10s$). The composition of each household is summarized in Table 3.1. We illustrate these databases with Figure 3.7 with one day of consumption for each appliance and Figure 3.8 with some examples of appliance signatures.

TraceBase is a collection of single-appliance measurements. It cannot be used for disaggregation purposes but for appliance recognition. It is a collection of measurements over one appliance during one day. In total, 11 types appliances are monitored, with several models for each appliances (from 1 to 12 models).

3.3.4 Problem formulation

As illustrated in Figure 3.7, the total disaggregation is hard since some large appliances are not monitored. In this thesis, we take interest in two sub-problems. The first sub-problem is the one of **appliance recognition**: is it possible to recognize the signature of a model of an appliance that has never been seen before?

Indeed, if such recognition is already impossible, load monitoring is even harder. For this purpose, we formulate the problem of time series classification and study two kinds of transferability: cross-house transferability and cross-dataset transferability. Indeed, REFIT and Trace Base datasets are respectively collected in the UK and in Germany. While European appliances are in general similar, we expect their signatures to be different from the ones in France.

The second problem is the one of **appliance daily consumption estimation**. It is a harder problem as it works on the aggregated load curve. Moreover, it is a regression problem, wich requires different techniques from classification. This problem is treated in Chapter 4.

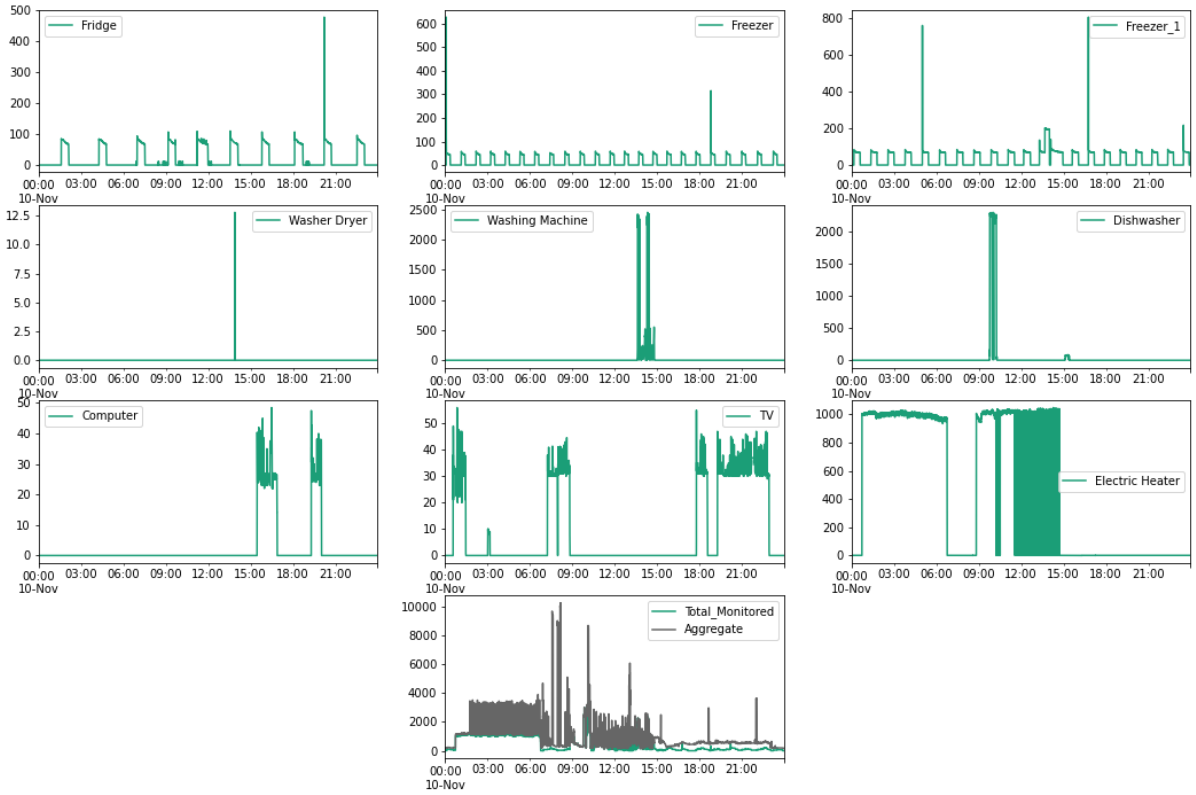


Figure 3.7: Examples of monitored consumption for one day in the REFIT Database (x-axis is in hours and y-axis is in W ; the house numbers correspond to the ones in the database as technical issues happened for other houses)

House 1	House 2	House 3	House 4	House 5	House 6	House 7
Fridge	Fridge	Toaster	Fridge	Fridge	Freezer	Fridge
Freezer	Washing Machine	Fridge	Freezer	Tumble Dryer	Washing Machine	Freezer
Freezer	Dishwasher	Freezer	Fridge	Washing Machine	Dishwasher	Freezer
Washer Dryer	TV	Tumble Dryer	Washing Machine	Dishwasher	Computer	Tumble Dryer
Washing Machine	Microwave	Dishwasher	Washing Machine	Computer	TV	Washing Machine
Dishwasher	Toaster	Washing Machine	Computer	Television	Microwave	Dishwasher
Computer	Hi-Fi	TV	TV	Microwave	Kettle	TV
TV	Kettle	Microwave	Microwave	Kettle	Toaster	Toaster
Electric Heater	Overhead Fan	Kettle	Kettle	Toaster	Computer	Kettle
House 8	**House 9**	**House 10**	**House 11**	**House 12**	**House 13**	**House 15**
Fridge	Fridge	Blender	Fridge	Fridge	TV	Fridge
Freezer	Washer Dryer	Toaster	Fridge	Unknown	Freezer	Tumble Dryer
Washer Dryer	Washing Machine	Freezer	Washing Machine	Unknown	Washing Machine	Washing Machine
Washing Machine	Dishwasher	Fridge	Dishwasher	Computer	Dishwasher	Dishwasher
Toaster	TV	Washing Machine	Computer	Microwave	Unknown	Computer
Computer	Microwave	Dishwasher	Microwave	Kettle	Network Site	TV
TV	Kettle	TV	Kettle	Toaster	Microwave	Microwave
Microwave	Hi-Fi	Microwave	Router	TV	Microwave	Hi-Fi
Kettle	Electric Heater	K Mix	Hi-Fi	Unknown	Kettle	Toaster
House 16	**House 17**	**House 18**	**House 19**	**House 20**	**House 21**	
Fridge	Freezer	Fridge	Fridge Freezer	Fridge	Fridge	
Fridge	Fridge	Freezer	Washing Machine	Freezer	Tumble Dryer	
Electric Heater	Tumble Dryer	Fridge	TV	Tumble Dryer	Washing Machine	
Electric Heater	Washing Machine	Washer Dryer	Microwave	Washing Machine	Dishwasher	
Washing Machine	Computer	Washing Machine	Kettle	Dishwasher	Food Mixer	
Dishwasher	TV	Dishwasher	Toaster	Computer	TV	
Computer	Microwave	Computer	Bread-maker	TV	Kettle	
TV	Kettle	TV	Games Console	Microwave	Vivarium	
Dehumidifier	TV	Microwave	Hi-Fi	Kettle	Pond Pump	

Table 3.1: Summary of REFIT households

3.4 Time Series Normalization for Invariant Appliance Recognition

3.4.1 Global and z-normalization

It is well known that pre-processing is crucial for time series analysis. Operations such as z-normalization, de-trending or de-seasonalizing are traditional pre-processing tools for time series analysis. Finding the appropriate pre-processing is not an easy task and generally depends on inner data characteristics. Each technique may discard some information and should be used with caution. For signature recognition, detrending and de-seasonalizing are not necessary. For Time Series Classification, z-normalization is classically used, including as a pre-processing for deep learning methods. But as was noted in [65], "this traditional pre-processing step should be further studied [...] since normalization is known to have a huge effect on DNNs' learning capabilities".

Firstly, we define two common normalization techniques for time series: standardization and min-max normalization. Each of them is broken down into per instance and global normalization. Here, $\mathbf{X} \in \mathbb{R}^{n \times T}$ is the full dataset where n is the number of samples and T the time series length, $\mathbf{Y} \in \mathbb{R}^{n \times C}$ is the label vector. $X_i = \{X_i^1, ..., X_i^T\}$ denotes the i^{th} element of $\mathbf{X}$.

Global min-max normalization (**GN**) normalizes the values of $\mathbf{X}$ according to its minimum and maximum converting it into the range $[0, 1]$. Typical issues would be out-of-sample minimum and maximum and outliers, which happens often in many time series applications. Global standardization (**GS**) standardizes the values of $\mathbf{X}$ according

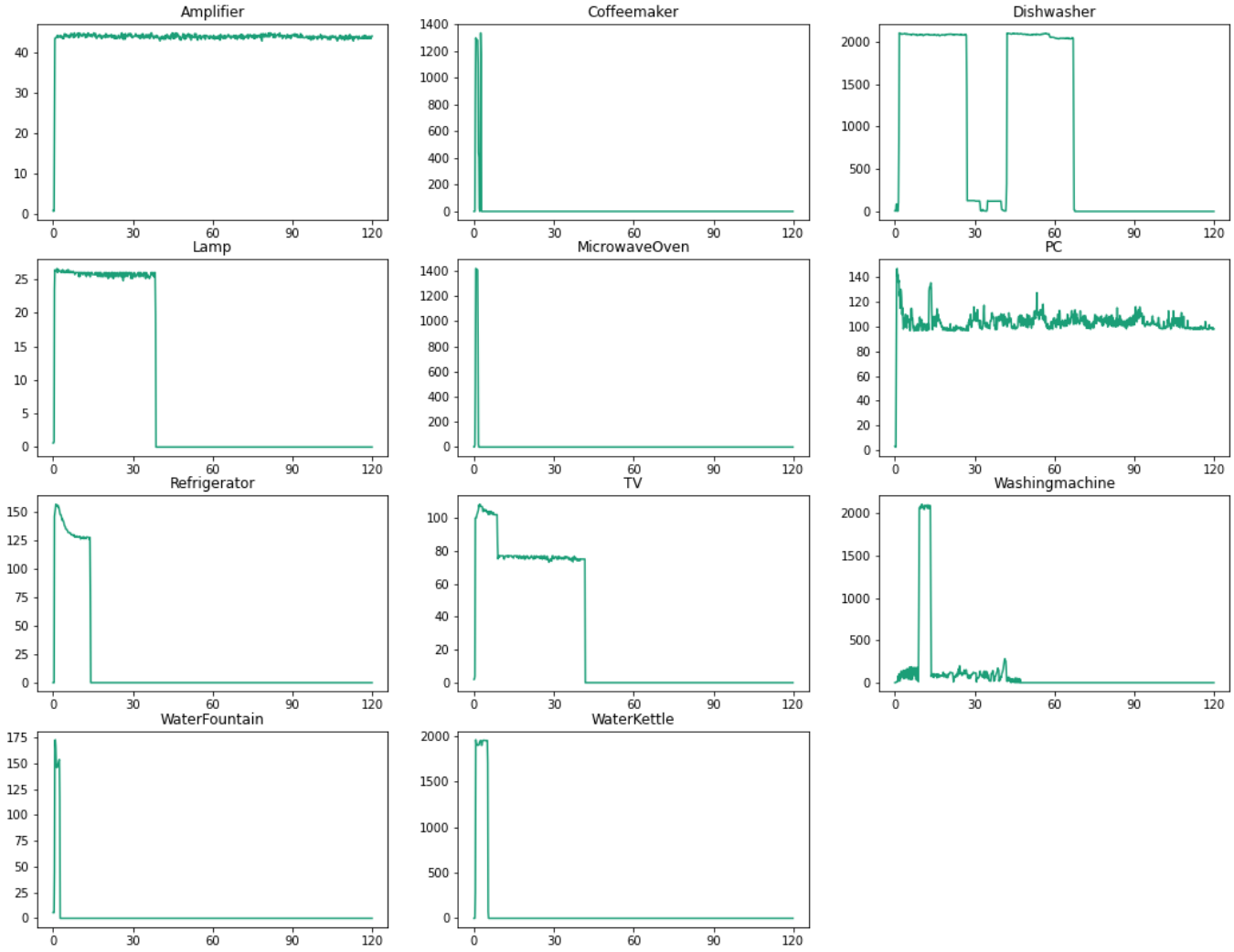
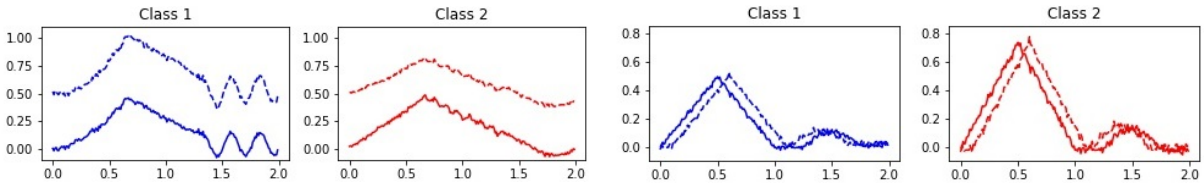


Figure 3.8: Examples of signatures from the Trace Base dataset (x-axis is in minutes, y-axis is in W). Signatures have been zero-padded to a length of 2 hours.



(a) Shape is discriminative

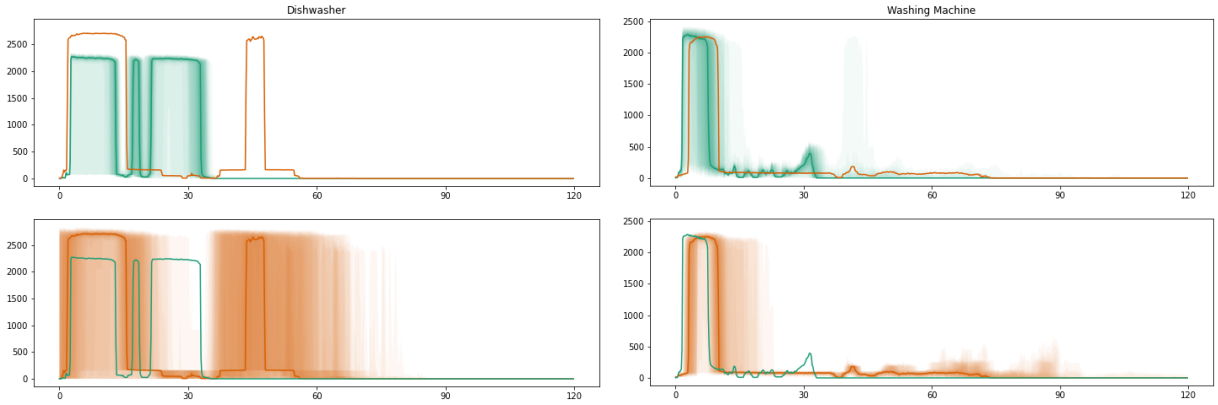
(b) Scale is discriminative

Figure 3.9: Toy examples

to its mean and standard deviation. This very common normalization is usually done per variable in order to give the same scale for each variable but for time series it would mean to normalize the data per time stamp, destroying temporal structure. Hence global transformations are a rescaling of data. Instance normalization differs since each time series of the dataset is normalized using its own statistics. We define Instance min-max Normalization (**IN**) and Instance Standardization (**IS**) in the Table 3.2.

	Min-Max	Standardization
Global	$GN(X_i^j) = \frac{X_i^j - min(\mathbf{X})}{max(\mathbf{X}) - min(\mathbf{X})}$	$GS(X_i^j) = \frac{X_i^j - mean(\mathbf{X})}{std(\mathbf{X})}$
Instance	$IN(X_i^j) = \frac{X_i^j - min(X_i)}{max(X_i) - min(X_i)}$	$IS(X_i^j) = \frac{X_i^j - mean(X_i)}{std(X_i)}$

Table 3.2: Normalization methods



(a) Dishwasher signatures: green and orange correspond to two different models of dishwashers

(b) Washing Machine signatures: green and orange correspond to two different models of washing machines

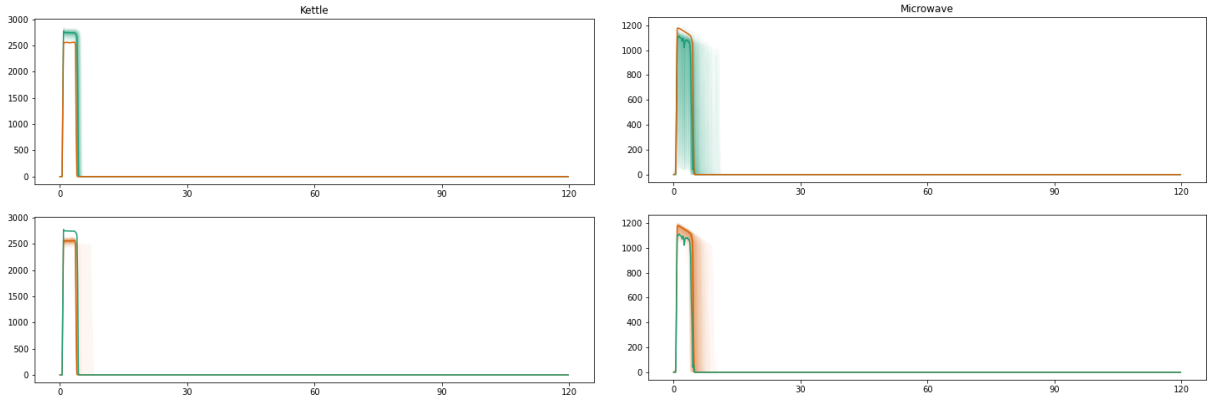
Figure 3.10: Dishwasher and Washing Machine Signature data: for each plot, the main line is the median consumption over all signatures. Other areas represent different percentiles of the distribution.

Global Transformations are simple re-scaling but they are useful for good training of neural networks. Most existing approaches use the instance standardization (also called z-normalization) to pre-process time series. For instance UCR archive included only z-normalized datasets until the 2018 update. The incentive behind this choice is that similarity between two time series can be meaningless without proper pre-processing in presence of an offset or a scale variation [146].

We argue that this choice might not be optimal for every domain, especially when the scale or the offset are discriminative. It can be seen on a toy example: in Figure 3.9a, without normalization, a euclidean or DTW-based classifier would not discriminate the classes properly: dotted time series is classified in the same class. With classes from Figure 3.9b, instance standardization has the opposite effect. One can notice that there is no obvious choice of normalization and for more complex data, balancing shape and scale information is challenging.

3.4.2 Normalization for appliance consumption

Using the previously mentioned notations, we analyze how it applies for appliance signature recognition. It was already mentioned in the seminal paper of Hart (1989) [85] that normalizing power measurements per appliance was useful to compare between different models of the same devices using voltage data. Since it is not available in our dataset, we simply use power measurements.



(a) Kettle signatures: green and orange correspond to two different models of kettles

(b) Microwave signatures: green and orange correspond to two different models of microwaves

Figure 3.11: Kettle and Microwave Signature data: for each plot, the main line is the median consumption over all signatures. Other areas represent different percentiles of the distribution.

Kelly and Knottenbelt (2015) [100] subtract each sequence by its own mean but divide each sequence by the standard deviation of the whole training set as a pre-processing step for neural networks. They already notice that it loses the information of the nominal power of an appliance which is crucial to other methods and assumed that "there are likely to be ways to have the best of both world".

Indeed, for some appliances, the nominal power is discriminative whereas for other appliances, only the shape matters. Figures 3.10 and Figure 3.11 illustrate this on two examples. For each pair (washing machine / dishwasher and kettle / microwave), we firstly plot the distribution of the signatures of two models of each appliance and then a comparison between appliances. For dishwashers and washing machines there is a higher variability but in general, dishwashers signatures are made of two cycles, whereas washing machine are made of only one cycle. The cycle length depends on the usage and the model of the appliance, while the nominal power is dependent on the model and varies between $2000W$ and $2500W$ for both appliances. For kettles and microwaves, they both have the same shape of signature, with a single short step. But the nominal power is different, as for kettles it is in general between $1500W$ and $2200W$, whereas for microwaves, it is around $1000W$.

The danger of instance normalization is visible on the example of kettles and microwaves, as both signatures are very similar as illustrated in Figure 3.11. But on the example of dishwashers and washing machines, normalizing would allow for better transferability between appliances, as the main difference between two models of dishwashers or washing machines is their nominal power. In order to combine both information and get the best of both worlds, we propose an ensembling model using deep learning.

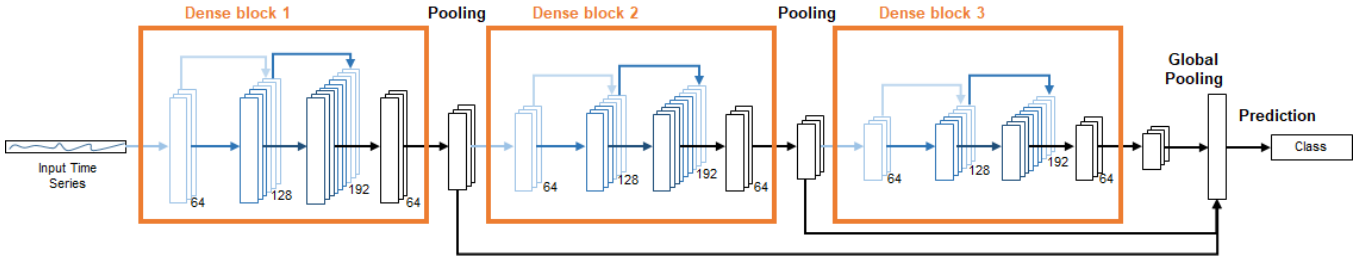


Figure 3.12: DenseNet: each block is made of successive convolutions and skip connections ended by a bottleneck convolution

3.4.3 Model

Base Model: DenseNet

Our model is based on convolutional neural networks presented in Section 3.2. Convolutional layers and intermediate pooling are successively applied in order to extract features at different scales before a global pooling operation and a fully connected layer that predicts a label. Inspired from the computer vision literature [90], we propose DenseNets for Time Series, illustrated in Figure 3.12.

DenseNets use skip connections from different levels in the network through concatenation. Namely, the k^{th} layer receives the outputs of some preceding layers as an input ; denoting z_k the output of the k^{th} layer and $[z_{k-m}, ..., z_{k-1}]$ the concatenation of the m previous layers, the output of the k^{th} layer is

$$z_k = f_k([z_{k-m}, ..., z_{k-1}])$$

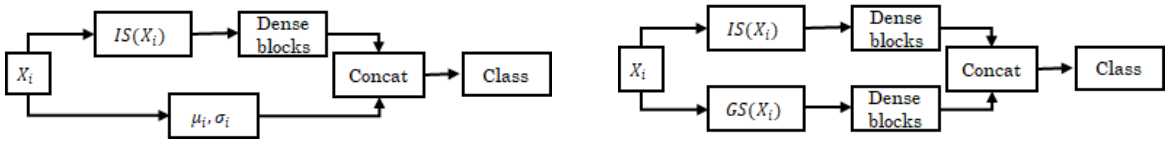
DenseNets allow to produce more complex information from layer outputs than ResNets. On the other hand, they tend to have larger layer inputs, due to the successive concatenations, depending on the number m of preceding layers being concatenated. In our architecture, we use bottleneck layers: after a dense block, a bottleneck layer brings back the number of inputs to the initial number of feature maps, as described on Figure 3.12. Hence for a fixed number of feature maps K and maximum m_{max} size of dense block, the number of inputs never exceeds $K \times m_{max}$.

Another novelty of our architecture is to explicitly feed features from different scales to the final predictor. Namely we concatenate the outputs of each dense block to create the input of the last fully connected layer. As the number of dense blocks is relatively small, the input size stays tractable.

DenseNets are comparable to ResNets, which are state of the art in time series classification, as they introduce skip connections. DenseNets showed better performance in the experiments but are also longer to train than ResNets.

Mixing scale and shape

In theory, it is rarely strictly necessary to standardize the inputs of a neural network and most pre-processing tricks are hard to analyze properly. In practice, standardization allows non-fittable networks to be fittable [107], as the gradients of most activation functions tend to be more informative for values close to 0. Hence, we always use at least a global normalization. Recently batch normalization proposed by Ioffe and Szegedy [95] has had a great impact on neural network training and works as a speedup technique for neural networks.



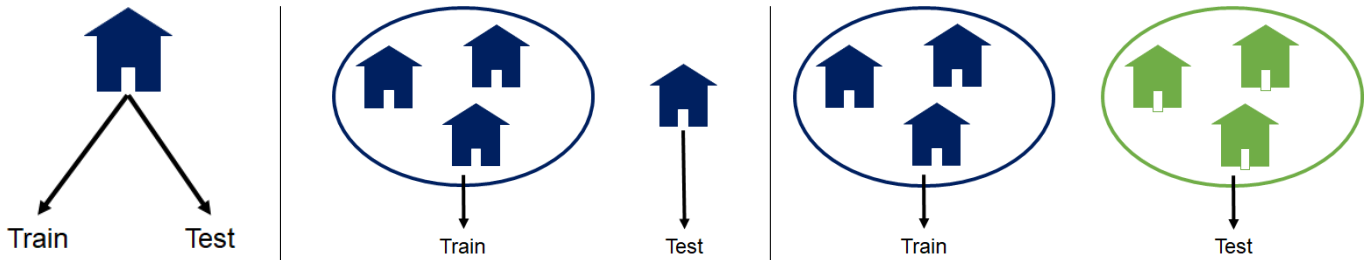
(a) FeatNet (b) EnsNormNet

Figure 3.13: Proposed architectures

The first solution is to create a new architecture with two entries as summarized in Figure 3.13a. One entry corresponds to the instance-normalized time series, which is passed into a convolutional architecture. At the fully-connected level, the output of the convolutional blocks is concatenated with the other entry, containing the scale information from the time series (mean and standard deviation for standardization ; minimum and maximum for min-max normalization).

The second solution is to create an architecture with different entries corresponding to the input time series, normalized and scaled differently for each entry. Each entry is passed into convolutional blocks with no weight sharing. The outputs are then concatenated into a fully connected layer that gives the final prediction. Creating separate channels with different information is similar to the Multi-Channel Neural Network introduced in [197].

3.5 Experiments on NILM Datasets



(a) Same House (b) Cross House (c) Cross Dataset

Figure 3.14: Overview of the different experiments

We present the results of the previous method in three scenario, illustrated in Figure 3.14:

- Same houses: the same devices are seen during training and testing
- Cross-houses: different devices are used for training and testing
- Cross-dataset: devices come from different datasets (different countries).

The first case is our baseline scenario. In the second case, we expect a shift in the distributions as the devices in the training houses are not the same as the ones in the test houses. Finally, in the cross-dataset scenatio, our objective is to study the transferability between countries and different experimental process.

3.5.1 Preprocessing and Methods

Preprocessing

Electric Data	REFIT	TraceBase
	Computer	Computer
Dishwasher	Dishwasher	Dishwasher
Kettle	Kettle	Kettle
Microwave	Microwave	Microwave
Oven		
	Toaster	
Tumble Dryer	Tumble Dryer	
	TV	TV
Washing Machine	Washing Machine	Washing Machine
Water Heater		

Table 3.3: Appliances for which signatures are extracted in each dataset.

For the REFIT and Electric Data datasets, we only have the measurements over the whole experiments (several months) for each appliance and need to extract each signature. In order to extract the signatures, we use a detector function to extract activations. It is based on several parameters:

- A minimum power to reach
- A minimum time window
- A minimum off period to split cycles correctly.

It is better to set these parameters specifically for each appliance. It must be noted that this function is not perfect, hence some untypical signatures can be extracted from raw data (extremely high consumption, very short spike, ...). For the REFIT dataset, we get satisfying results after simply removing outliers based on Local Outlier Factor. For Electric Data, it must be noted that the final database has been created using manual intervention. Firstly, some signatures are wrongly annotated in the experiment and had to be changed manually. Secondly, some signatures are curated with outlier detection, but we still noticed some remaining untypical signatures. Other methods for signature extraction are being developed at EDF but were not available during our work.

In the end, we extracted signatures for different appliances in each dataset as summarized in Table 3.3. Water heater is particularly present in France, while in the UK and in Germany gas heating is more popular. Moreover, our architecture requires time series to be of the same length. For this reason, we use right zero-padding to make signatures 2 hours long, which gives 720 points with a $10s$ sampling. We also tried architectures that do not require same-length time series such as recurrent networks or Fully Convolutional Networks, but results are worse.

For the TraceBase dataset, signatures are already extracted. We do not know the house of origin of each device, so we artificially created 7 house splits by gathering some devices together. The database is quite small with ~ 1000 signatures in total and with only 20 signatures for some appliance models. One can note, that even for REFIT and Electric Data, in the case of appliance recognition, the cross-house framework could be extended considering any combination of devices coming from different houses (for instance, train on some devices from houses 1 and 2 and test on some devices from houses 3 and 4), as long as we keep only one model per appliance.

Compared Methods

In the experiments, we compared different methods:

1. **RF:** Feature extraction + RandomForest
2. **GS** DenseNet with Global Standardization
3. **GN** DenseNet with Global Min-Max Normalization
4. **IS** DenseNet with Instance Standardization
5. **IN** DenseNet with Instance Min-Max Normalization
6. **IN-Feat** FeatNet with Instance Min-Max Normalization
7. **Ens** EnsNormNet with Min-Max Normalization
8. **IN-DA** Domain Adversarial DenseNet with Instance Min-Max Normalization
9. **GN-DA** Domain Adversarial DenseNet with Global Min-Max Normalization
10. **Ens-DA** Domain Adversarial EnsNormNet with Min-Max Normalization

The first method is a non deep learning method. In **RF**, 113 statistical features from the whole time series and windows are extracted and feeded in a random forest. The feature extraction is made on signatures of different lengths (we do not need different padding). The methods 3 to 8 are described in the previous section. The last three methods are only used for the cross-house and cross-dataset experiments as they are adaptation techniques. They

correspond to the adversarial scheme described in Section 2.3.2: at the dense layers in our architecture, an additional branch is added to learn a discriminator between domains. Other methods were tried, such as Discriminative Dictionary Learning [102], but gave very poor results.

The DenseNet architecture is kept the same for every neural network based method. It is made of 3 dense blocks, each of them composed of 3 convolutional layers with 64 filters of size 7, 5 and 3. We tried other architectures but we found that with more than 3 convolutional layers, the network could not fit anymore. We ran each experiment 10 times with different seeds for weight initialization and we report only mean F1-scores with standard deviations.

Layers	Input shape	Output shape	Filter shape
Input	Time series of length l		
Conv (1)	$l \times 1$	$l \times 64$	1×7
Conv (2)	$l \times 64$	$l \times 64$	64×5
Conv (3)	$l \times 128$	$l \times 64$	128×3
Conv (4)	$l \times 192$	$l \times 64$	192×3
Pooling (1)	$l \times 64$	$l/2 \times 64$	2
Conv (5)	$l/2 \times 64$	$l/2 \times 64$	64×7
Conv (6)	$l/2 \times 64$	$l/2 \times 64$	64×5
Conv (7)	$l/2 \times 128$	$l/2 \times 64$	128×3
Conv (8)	$l/2 \times 192$	$l/2 \times 64$	192×3
Pooling (2)	$l/2 \times 64$	$l/4 \times 64$	2
Conv (9)	$l/4 \times 64$	$l/4 \times 64$	64×7
Conv (10)	$l/4 \times 64$	$l/4 \times 64$	64×5
Conv (11)	$l/2 \times 128$	$l/4 \times 64$	128×3
Conv (12)	$l/4 \times 192$	$l/4 \times 64$	192×3
Pooling (3)	$l/4 \times 64$	$l/8 \times 64$	2
Merge	Pooling (1)+(2)+(3)		
Dense	$56l$	n_{class}	$56l$

Table 3.4: DenseNet architecture used in both experiments

Layers	Input shape	Output shape	Filter shape
Input (1)	Instance-standardized time series		
Conv (1)	$l \times 1$	$l \times 64$	1×7
Conv (2)	$l \times 64$	$l \times 64$	64×5
Conv (3)	$l \times 128$	$l \times 64$	128×3
Conv (4)	$l \times 192$	$l \times 64$	192×3
Pooling (1)	$l \times 64$	$l/2 \times 64$	2
Conv (5)	$l/2 \times 64$	$l/2 \times 64$	64×7
Conv (6)	$l/2 \times 64$	$l/2 \times 64$	64×5
Conv (7)	$l/2 \times 128$	$l/2 \times 64$	128×3
Conv (8)	$l/2 \times 192$	$l/2 \times 64$	192×3
Pooling (2)	$l/2 \times 64$	$l/4 \times 64$	2
Conv (9)	$l/4 \times 64$	$l/4 \times 64$	64×7
Conv (10)	$l/4 \times 64$	$l/4 \times 64$	64×5
Conv (11)	$l/2 \times 128$	$l/4 \times 64$	128×3
Conv (12)	$l/4 \times 192$	$l/4 \times 64$	192×3
Pooling (3)	$l/4 \times 64$	$l/8 \times 64$	2
Input (2)	μ_i, σ_i		
Merge	Pooling (1)+(2)+(3) + Input (2)		
Dense	$56l + 2$	n_{class}	$56l + 2$

Table 3.5: FeatNet architecture used in both experiments

Layers	Input shape	Output shape	Filter shape	Layers	Input shape	Output shape	Filter shape
Input (1)	Global-Standardized TS			Input (2)	Instance-Standardized TS		
Conv (1)	$l \times 1$	$l \times 64$	1×7	Conv (13)	$l \times 1$	$l \times 64$	1×7
Conv (2)	$l \times 64$	$l \times 64$	64×5	Conv (14)	$l \times 64$	$l \times 64$	64×5
Conv (3)	$l \times 128$	$l \times 64$	128×3	Conv (15)	$l \times 128$	$l \times 64$	128×3
Conv (4)	$l \times 192$	$l \times 64$	192×3	Conv (16)	$l \times 192$	$l \times 64$	192×3
Pooling (1)	$l \times 64$	$l/2 \times 64$	2	Pooling (4)	$l \times 64$	$l/2 \times 64$	2
Conv (5)	$l/2 \times 64$	$l/2 \times 64$	64×7	Conv (17)	$l/2 \times 64$	$l/2 \times 64$	64×7
Conv (6)	$l/2 \times 64$	$l/2 \times 64$	64×5	Conv (18)	$l/2 \times 64$	$l/2 \times 64$	64×5
Conv (7)	$l/2 \times 128$	$l/2 \times 64$	128×3	Conv (19)	$l/2 \times 128$	$l/2 \times 64$	128×3
Conv (8)	$l/2 \times 192$	$l/2 \times 64$	192×3	Conv (20)	$l/2 \times 192$	$l/2 \times 64$	192×3
Pooling (2)	$l/2 \times 64$	$l/4 \times 64$	2	Pooling (5)	$l/2 \times 64$	$l/4 \times 64$	2
Conv (9)	$l/4 \times 64$	$l/4 \times 64$	64×7	Conv (21)	$l/4 \times 64$	$l/4 \times 64$	64×7
Conv (10)	$l/4 \times 64$	$l/4 \times 64$	64×5	Conv (22)	$l/4 \times 64$	$l/4 \times 64$	64×5
Conv (11)	$l/2 \times 128$	$l/4 \times 64$	128×3	Conv (23)	$l/2 \times 128$	$l/4 \times 64$	128×3
Conv (12)	$l/4 \times 192$	$l/4 \times 64$	192×3	Conv (24)	$l/4 \times 192$	$l/4 \times 64$	192×3
Pooling (3)	$l/4 \times 64$	$l/8 \times 64$	2	Pooling (6)	$l/4 \times 64$	$l/8 \times 64$	2
Merge	Pooling (1)+(2)+(3)+(4)+(5)+(6)						
Dense	Class prediction						

Table 3.6: EnsNormNet architecture: dense blocks are applied in parallel before being merged

3.5.2 Same House

When training and testing on signatures coming from the same house, there is very little shift in the signatures. Overall, the problem is quite easy and every algorithm performs well. We report the macro-F1 score on Electric Data database in Table 3.7. We ran the experiments 10 times with different initial weights. In this database, we only consider large appliances, hence it is quite easy to recognize them for every method. In general, the few mislabeled signatures are outliers. We did not notice significant differences between min-max normalization and standardization.

Electric Data							
Method	**RF**	**GS**	**GN**	**IS**	**IN**	**IS-Feat**	**Ens**
F1 Score	**97,12 (0,04)**	97,04 (0,11)	96,99 (0,09)	96,12 (0,16)	96,18 (0,07)	96,23 (0,18)	**97,05 (0,16)**

Table 3.7: Same house: Macro F1 score (%) for different methods with standard deviation over 10 runs (different initial weights) on REFIT dataset

For REFIT and Trace Base, similar results are obtained (see Table 3.8). The only exception is for neural network using instance normalization for some devices such as kettle and microwaves. These results indicate that appliance recognition is easy when training and testing on the same appliance models and the same users.

REFIT Dataset							
Method	**RF**	**GS**	**GN**	**IS**	**IN**	**IS-Feat**	**Ens**
F1 Score	**98,42 (0,01)**	**98,47 (0,11)**	**98,52 (0,12)**	96,33 (0,21)	96,28 (0,17)	98,32 (0,24)	**98,57 (0,14)**

Table 3.8: Same house: F1 score (%) for different methods with standard deviation over 10 runs on REFIT dataset

TraceBase Dataset							
Method	**RF**	**GS**	**GN**	**IS**	**IN**	**IS-Feat**	**Ens**
F1 Score	**98,42 (0,01)**	**98,47 (0,11)**	**98,52 (0,12)**	96,33 (0,21)	96,28 (0,17)	98,32 (0,24)	**98,57 (0,14)**

Table 3.9: Same house: F1 score (%) for different methods with standard deviation over 10 runs on REFIT dataset

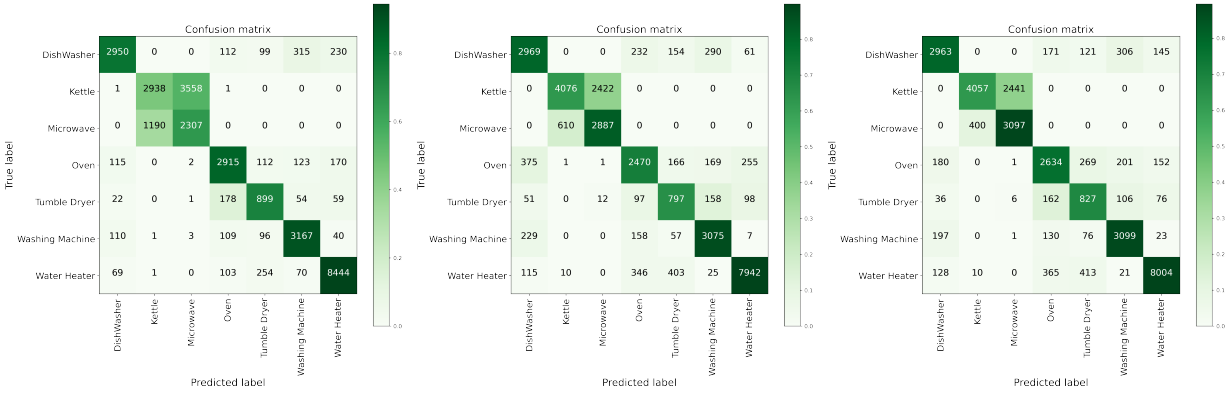
3.5.3 Cross-House Results

In this case, we sequentially consider one house as the test house and merge every other as a training house. We report in Figure 3.16 the whole confusion matrix for REFIT data for **GN**, **IN** and **Ens**. Results for ElectricData and TraceBase are represented in Figure 3.15 and Figure 3.17 respectively. Here, we clearly observe the gain of our ensemble technique. The observations made in Section 3.4.2 are confirmed with our experiments, as the Instance normalization methods tend to under-perform for appliances such as kettles, microwaves, TVs, and computers. On the other hand, when keeping scale information, some signatures from washing machines tend to be misclassified, especially for one house where the dishwasher has a high nominal power.

On Electric Data, similar observations can be made. Even with global normalization, the kettles and microwaves are still hard to discriminate. Indeed, kettles in this dataset tend to have a lower nominal power and are often mixed

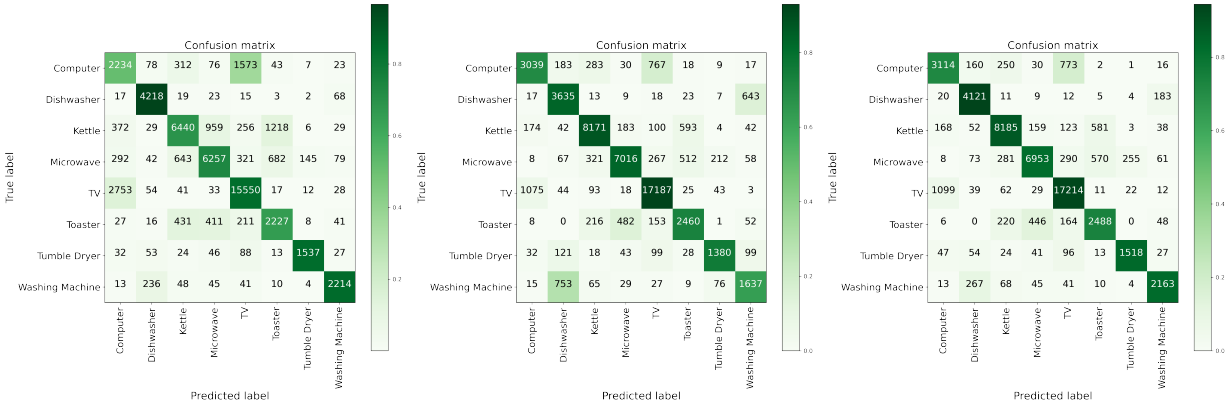
with microwaves.

On TraceBase data, the results are similar with every method, although once again, our EnsNormNet gives the best results of all methods. TraceBase is only made of very clean signatures and has a lower variety of appliance models in general. Hence, there is a high homogeneity between appliances and a low shift between houses.



(a) Instance Normalization (b) Global Normalization (c) EnsNormNet

Figure 3.15: Confusion Matrix for Electric Data in the cross-house appliance recognition experiment

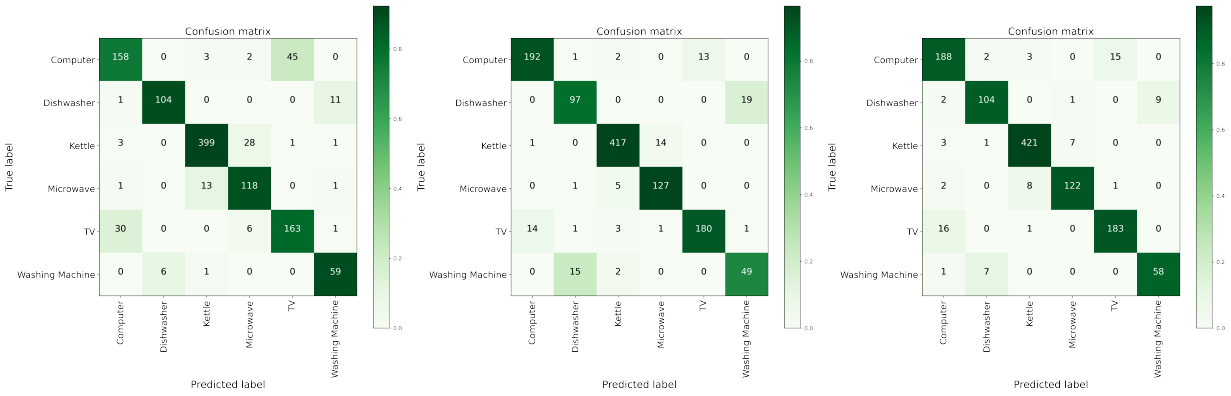


(a) Instance Normalization (b) Global Normalization (c) EnsNormNet

Figure 3.16: Confusion Matrix for REFIT Data in the cross-house appliance recognition experiment

In Table 3.11, Table 3.10 and Table 3.12, we present the macro-F1 score for every method on ElectricData, REFIT and TraceBase respectively. We expected the FeatNet method to perform as well, but it gives very similar results to Global Normalization, which could mean that the network gives too much importance to the scale information. Similarly, the random forest model underperforms compared to neural nets-based methods. Other methods such as dictionary learning or Nearest Neighbours with Dynamic Time Warping are both underperforming compared to the Random Forest model.

Another thing to notice is that the Domain Adversarial Training does not bring any improvement to the classification performance. In fact, on Electric Data it even brings down the classification performance. Since it is unsupervised adaptation, it tended to match representations without any knowledge about the label, which com-



(a) Instance Normalization (b) Global Normalization (c) EnsNormNet

Figure 3.17: Confusion Matrix for TraceBase Data in the cross-house appliance recognition experiment

pletely mixed together kettle and microwaves and even large appliances. We could improve on these results by selecting only houses similar to the target houses but the similarity measure required supervision (knowledge about the target appliances). This motivates the idea of source selection which we develop in the next chapter.

Electric Data				
RF	**GS**	**GN**	**IS**	**IN**
68,83 (0,06)	75,13 (0,37)	75,10 (0,47)	74,38 (0,21)	74,74 (0,29)
IN-Feat	**Ens**	**IN-DA**	**GN-DA**	**Ens-DA**
75,88 (0,43)	**78,45 (0.33)**	66,24 (1,00)	69,87 (1,12)	68,32 (1,52)

Table 3.10: Cross-house: F1 score (%) for different methods with standard deviation over 10 runs on Electric Data dataset

REFIT Dataset				
RF	**GS**	**GN**	**IS**	**IN**
73,12 (0,08)	78,37 (0,63)	77,37 (0,58)	75,69 (0,89)	75,48 (0,76)
IN-Feat	**Ens**	**IN-DA**	**GN-DA**	**Ens-DA**
76,11 (0,97)	**83,39 (0,54)**	75,28 (0,87)	76,12 (0,68)	81,14 (1,02)

Table 3.11: Cross-house: F1 score (%) for different methods with standard deviation over 10 runs on REFIT dataset

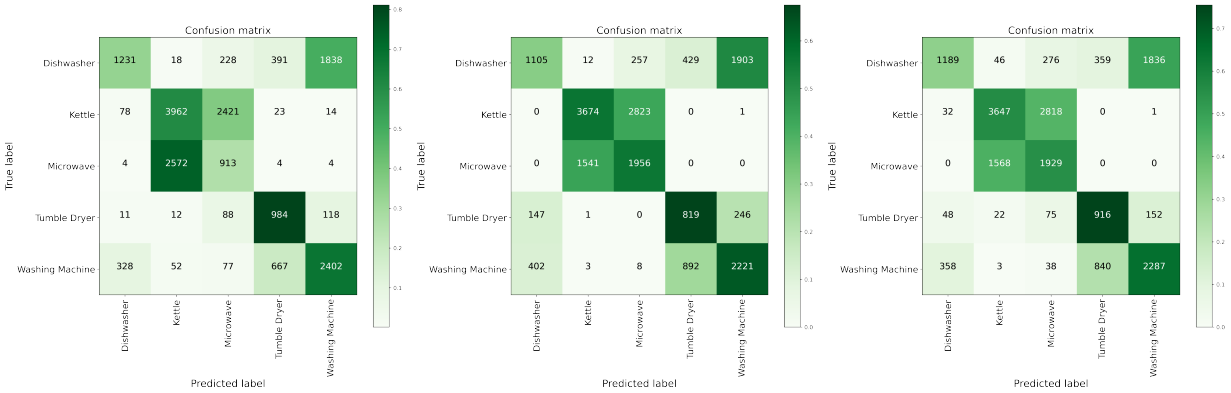
TraceBase Dataset				
RF	**GS**	**GN**	**IS**	**IN**
87,28 (0,21)	88,12 (0,31)	88,32 (0,34)	85,67 (0,29)	85,30 (0,28)
IN-Feat	**Ens**	**IN-DA**	**GN-DA**	**Ens-DA**
87,99 (0,41)	**91,50 (0,30)**	85,45 (0,43)	86,80 (0,51)	90,08 (0.37)

Table 3.12: Cross-house: F1 score (%) for different methods with standard deviation over 10 runs on TraceBase dataset

3.5.4 Cross-Dataset Results

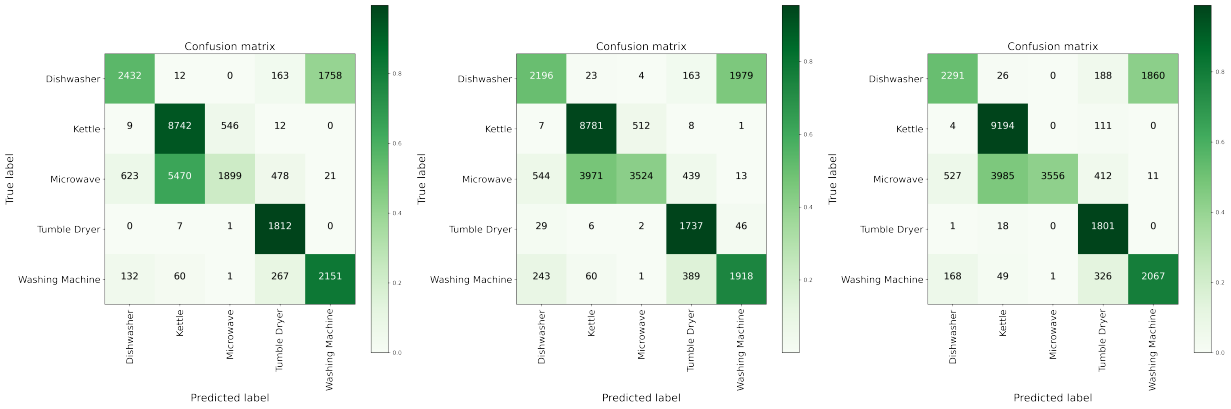
Finally, we study if appliance recognition is transferable between datasets coming from different countries. Here, we only consider European countries where appliances are similar. For each experiment, we only keep the appliances

available in every dataset. We merge signatures coming from every houses for the source dataset and applied on the target dataset. We report the confusion matrix for the REFIT $\rightarrow$ ElectricData and ElectricData $\rightarrow$ REFIT scenarios in Figure 3.18 and Figure 3.19. We note that models learnt on TraceBase give very poor results on Electric Data and REFIT, maybe due to the lack of data for appliances common to these datasets in the TraceBase dataset (~ 1000 signatures).



(a) Instance Normalization (b) Global Normalization (c) EnsNormNet

Figure 3.18: REFIT $\rightarrow$ Electric Data experiment



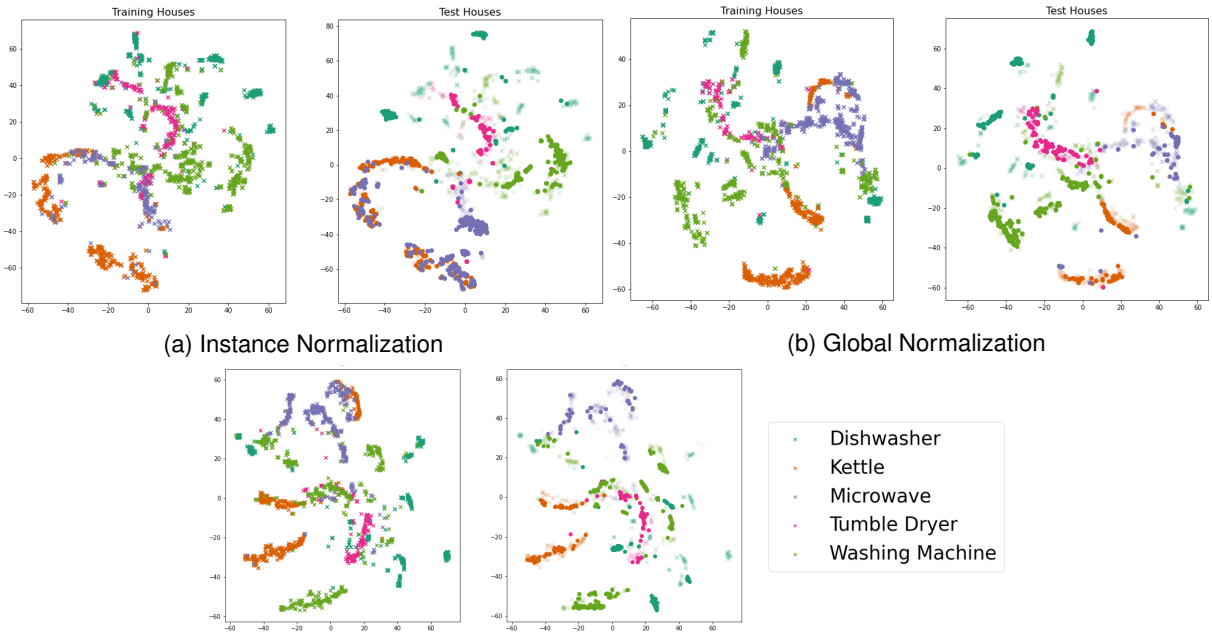
(a) Instance Normalization (b) Global Normalization (c) EnsNormNet

Figure 3.19: Electric Data $\rightarrow$ REFIT experiment

The results are poor compared to the cross-house scenario (note that the F1 scores are not directly comparable as the classes are different). Our ensembling method still gives the best results but in total the recognition accuracy is around 54%. The main issue is with small appliances like kettle and microwave: in the UK, kettles tend to have a higher nominal power but it is not the case in Electric Data. This is in line with a recent study [61] that showed that transferring between countries is not easy. Once again, the unsupervised domain adaptation methods does not give any improvement.

3.5.5 Discussion

A first conclusion is that, as expected, both scale and shape information matter for appliance recognition. Hence, our ensembling method takes the best of both worlds and allows better generalization. While not directly a Transfer Learning method, it still allows to reduce the shift between different models of the same appliances. We visualize it on Figure 3.20, where we show a visualization of the features learnt by the neural network. In the first subfigure, the network was learnt using Instance Normalization, in the second using Global Normalization and in the third, we use our Ensembling Network. We represent the features of appliances coming from a house not seen during training.



(a) Instance Normalization

(b) Global Normalization

(c) EnsNormNet

Figure 3.20: TSNE Representation of the latent variables obtained by learning a DenseNet using different normalizations: first column is the latent training variables and second column is the latent variables of unseen houses (test) on REFIT data

It is possible to conclude that with instance normalization, the network can separate training data but struggles with unseen data, especially for the microwave/kettle separation. The shifts are smaller using Global Normalization and we see that EnsNormNet manages to bet the best of both worlds. Here, the adaptation is very specific to our problem at hand. In a way, using two channels with different information is also related to data augmentation. Indeed, by feeding different information on the time series to the network, we perform implicit data augmentation, which is known to be good for better generalization. Finally, we also noticed that the domain adversarial methods did not give any improvement on the classification performance.

As we want to reduce shifts between houses, we also introduce other normalization methods, for instance by standardizing using the mean and the standard deviation of each appliance in each house. In that way, the distribution of power measurements of each appliance would have the same first two moments. The results are very

similar to the instance normalization approach as it completely ignored the nominal power of the appliance. We tried using instance weighting as described in Section 2.3.1 but it gave poor results.

It must be noted that such experiments are hard to conduct as it is not clear what a signature is. Here, we use a function depending on different parameters which is imperfect. We know that by setting stricter parameters, we can get a cleaner database where appliance recognition is easier but our database is incomplete, as some signatures be missing. The zero-padding is also questionable.

It would also be possible to use a hierarchical structure on the classes, such as learning a first predictor to separate large appliances (washing machine, dishwasher, ...) from small ones (TV, computer, kettles, ...) and two predictors for each categories. When we tried such method, the results are not better than in a general model. Finally, additional experiments are conducted on the UCR suite where we show that **Ens** always outperform methods based on only one normalization. These results are reported in Appendix ...

3.6 Conclusion

In this chapter, we presented different representations for time series with an emphasis on deep learning methods. We saw that deep learning methods, in particular convolutional neural networks, are very efficient tools for time series classification. This feature extraction is key for transfer learning as the transfer is very dependent on the feature space. As detailed in Chapter 2 many successful methods extract transferable features from the data. Hence one could perform a feature extraction on time series and then apply one of the Transfer Learning methods seen previously. But it would be sub-optimal as we expect that driving the feature extraction with the goal of Transfer Learning leads to more transferable features.

In this part, we proposed a method for appliance recognition. We observe that there is a gap in the performance between same-house appliance recognition and cross-house appliance recognition. In order to reduce this gap, we proposed a method that takes information from both shape and scale of time series and experimentally show that it allows to reduce the shift between appliances models for better transferability between houses. When trying domain adaptation techniques, no improvement was observed.

Nevertheless, our method is very field specific. Even for the more general problem of appliance consumption estimation, it is not always applicable. Domain adaptation methods presented in Chapter 2 do not integrate field specific information, whereas our ensemble methods does. As described in Figure 3.4, appliance recognition is the first sub-problem of Non Intrusive Load Monitoring. Our method is useful when collecting new data: indeed, during the collection of Electric Data datasets, a lot of manual re-labelling was necessary for mis-annotated data or residents switching appliances between plugs. We can automate this process with this method for future data acquisition. Moreover for the real problem of NILM, in general the aim is not to discriminate between small appliances. It would still be interesting to apply the method using a clean database for training, as new methods are currently

tested to extract signatures from Electric Data.

While our method displays good experimental results, there are still some open questions for time series classification using deep neural networks. We saw that because of the lack of a large benchmark database for time series, finding a very general architecture for time series is a hard issue. We argue that the main effort to solve this problem is the collection and preparation of such a large dataset. Moreover, deep neural networks are not very interpretable. For industrial applications this often is an issue, as practitioners are reluctant to use them. Research on more interpretable neural networks is in progress [194] and this will be a key factor for better industrialization of such methods.

Chapter 4

Domain adaptation with multiple sources in regression

Contents

The sub-problem of appliance classification presented previously shows that while a shift existed between houses, it ishard to reduce it using typical domain adaptation methods. Based on this observation we develop a framework for the problem of appliance consumption estimation, *ie* predicting the consumption of one appliance over a time window given the whole house consumption illustrated in Figure 4.1. We formulate this problem as a multiple source domain adaptation problem for regression, where different houses are considered as multiple sources to a different target house.

Chapter 2 highlights that domain adaptation for regression requires a specific framework. The main line of our work follows the idea of discrepancy minimization [121], solved using kernel based methods. The last Chapter shows that neural networks are powerful tools to learn features for consumption data. Hence, we develop a new framework for adversarial domain adaptation based on neural networks for regression.

This chapter introduces an extension to the original discrepancy first introduced by Mansour et al. (2009) [121]: hypothesis-discrepancy. Based on this new measure of dissimilarity between domains, we prove a general bound for domain adaptation encompassing both regression and classification tighter than the one with discrepancy. We extend our method to domain adaptation with multiple sources, and derive an algorithm that can select the most informative sources and adapt to the target at the same time. Our hypothesis-discrepancy based algorithm appears to unify different frameworks of domain adversarial training of neural networks. Finally, we study the applicability of our method on Electric Data for NILM and other kinds of data.

This chapter is partly based on the work published in an international conference and a pre-print:

- G. Richard, A. de Mathelin, G. Hébrail, M. Mougeot and N. Vayatis. "Unsupervised Multi-Source Domain Adaptation for Regression." Joint European Conference on Machine Learning and Knowledge Discovery in Databases 2020 (ECML 2020)

- A. de Mathelin, G. Richard, M. Mougeot, and N. Vayatis. "Adversarial Weighting for Domain Adaptation in Regression." arXiv preprint arXiv:2006.08251. (Pre-print)

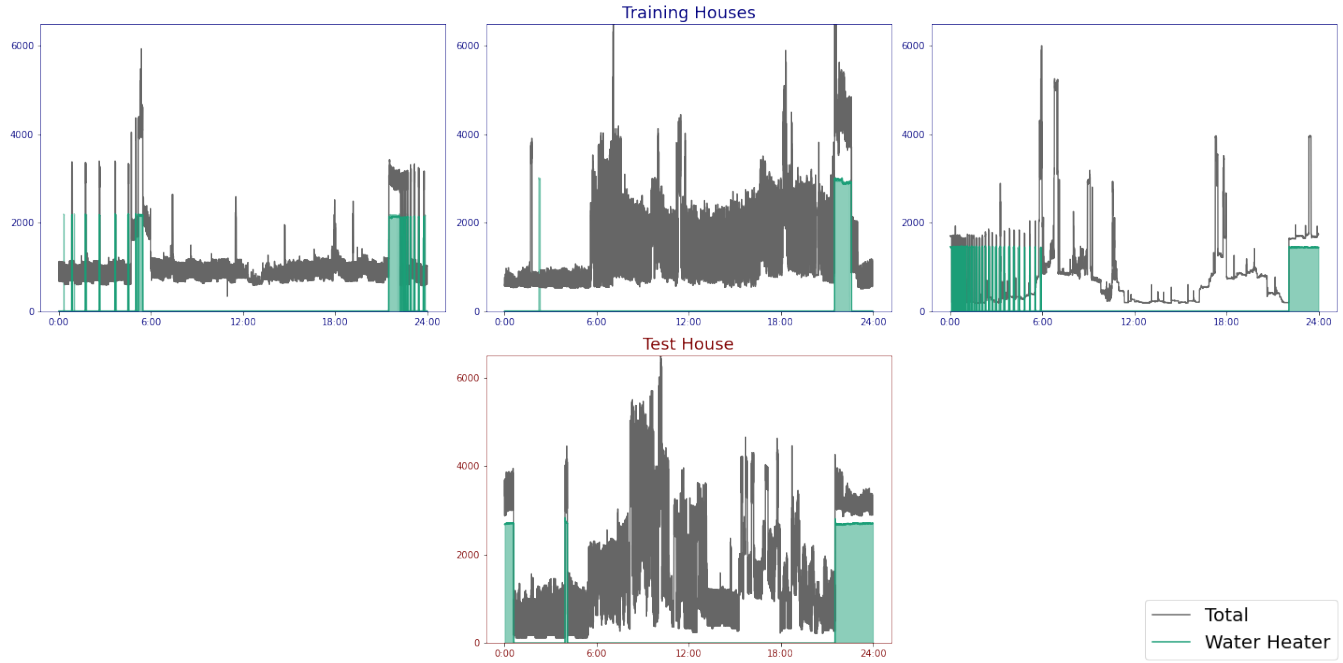


Figure 4.1: Water heater consumption estimation: input is the whole consumption (gray curve), variable to predict is the whole Water Heater consumption (green area)

4.1 Domain Adversarial Learning with $\mathcal{H}$-divergence

4.1.1 Literature review

We consider two domains $\mathcal{D}_S = \{\mathcal{X}, P_S(X_S)\}$ and $\mathcal{D}_T = \{\mathcal{X}, P_T(X_T)\}$ and two tasks $\mathcal{T}_S = \{\mathcal{Y}, P_S(Y_S|X_S)\}$ and $\mathcal{T}_T = \{\mathcal{Y}, P_T(Y_T|X_T)\}$ as previously defined in Chapter 2. We note f_S and f_T the source and target labelling functions, $\mathcal{H}$ a hypothesis class and $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ a loss function. We respectively note $\mathbf{R_S}(h)$ and $\mathbf{R_T}(h)$ the source and target risks of an hypothesis $h \in \mathcal{H}$.

As mentioned in Section 2.3.2, adversarial domain adaptation has been introduced by Ganin et al. (2016) [72]. The main idea is to learn transferable discriminative features using a neural networks optimizing two objectives. Considering a classification task with $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} \in \{0, 1\}$, authors introduce a neural network feature extractor $\phi : \mathcal{X} \to \mathbb{R}^m$, a neural network predictor $h : \mathbb{R}^m \to \mathcal{Y}$ and a neural network discriminator $D : \mathbb{R}^m \to \{S, T\}$. The predictor aims to learn to classify the input data while the discriminator aims to discriminate between domains. Namely, considering a labeled source sample $\mathcal{S}_S = \{(x_S^{(1)}, y_S^{(1)}), ..., (x_S^{(n_s)}, y_S^{(n_S)})\}$ of size n_S and an unlabeled target sample $\mathcal{S}_T = \{x_T^{(1)}, ..., x_T^{(n_T)}\}$ of size n_T, two losses are introduced:

- $\mathcal{L}_y = \sum_{i=1}^{n_S} L\left(h(\phi(x_S^{(i)})), y_S^{(i)}\right)$
- $\mathcal{L}_d = \sum_{i=1}^{n_S} L\left(D(\phi(x_S^{(i)})), 0\right) + \sum_{i=1}^{n_T} L\left(D(\phi(x_T^{(i)})\right), 1)$

where L is a classification loss, such as cross entropy. Finally, the objective is to minimize

$$\min_{h,\phi} \max_{D} \mathcal{L}_y + \lambda \mathcal{L}_d \tag{4.1}$$

This is done using the traditional feed-forward backpropagation and a gradient reversal layer between the feature extractor and the discriminator.

This idea is directly motivated by the $\mathcal{H}\Delta\mathcal{H}$-divergence[1] introduced by Ben-David et al. (2010) [25]. In this seminal paper authors already proposed to approximate $d_{\mathcal{H}\Delta\mathcal{H}}$ by $d_{\mathcal{H}}$ and try to find a representation minimizing $d_{\mathcal{H}}$. Moreover, $d_{\mathcal{H}}$ can be estimated using the loss $\mathcal{L}_d$ defined above. Hence, looking back at Equation 2.16 recalled below, the target risk is controlled by minimizing those two losses.

$$\mathbf{R}_T(h) \leq \mathbf{R}_S(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(p_S, p_T) + \lambda \tag{2.16}$$

From there, other adversarial methods have been proposed. ADDA, proposed by Tzeng et al. (2017) [175] is similar to DANN but uses asymmetric features (different feature extractors for source and target data). Another method of interest is Maximum Classifier Discrepancy introduced by Saito et al. (2018) [158]. In this method, authors propose to (a) learn two classifiers h and h' to minimize the source risk (b) maximize the discrepancy between those classifiers in the target domain (c) minimize the discrepancy induced by those classifiers in the feature extractors.

This is more closely related to $d_{\mathcal{H}\Delta\mathcal{H}}(P_S, P_T) = \sup_{h,h'} |\mathbb{E}_{x\sim P_S}[I(h(x) \neq h'(x)] - \mathbb{E}_{x\sim P_T}[I(h(x) \neq h'(x)]|$ as pointed out in [158]. Here, assuming h and h' close on the source dataset (done by step (a) of their algorithm), $\mathbb{E}_{x\sim P_S}[I(h(x) \neq h'(x)] \simeq 0$. Hence, maximizing the discrepancy on the target dataset induced by $\mathbb{E}_{x\sim P_T}[I(h(x) \neq h'(x)]$ gives an estimation of $\mathcal{H}\Delta\mathcal{H}$. It can be intepreted as reducing $\mathcal{H}\Delta\mathcal{H}$ to the set of "good" predictors for the source data.

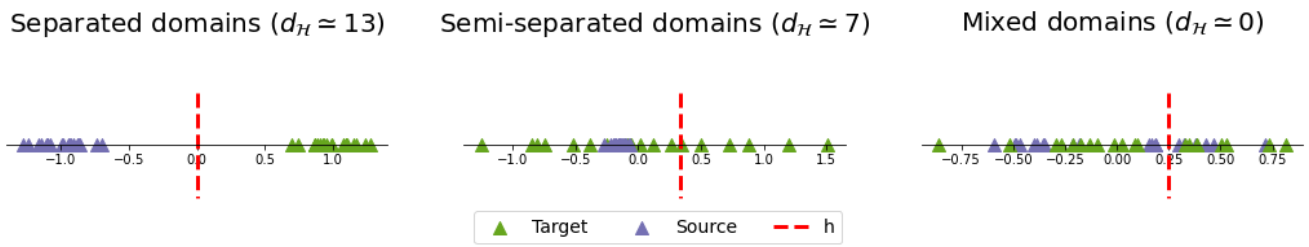
4.1.2 Limits of Domain Adversarial Adaptation in Regression with $\mathcal{H}$-divergence

Both previous methods base their analysis on the specific problem of binary classification. Moreover, they both minimize a lower bound on $d_{\mathcal{H}\Delta\mathcal{H}}$ as $d_{\mathcal{H}} \leq d_{\mathcal{H}\Delta\mathcal{H}}$. Chapter 2 also presented that the discrepancy is able to generalize $d_{\mathcal{H}\Delta\mathcal{H}}$ to more general classes of loss and predictors.

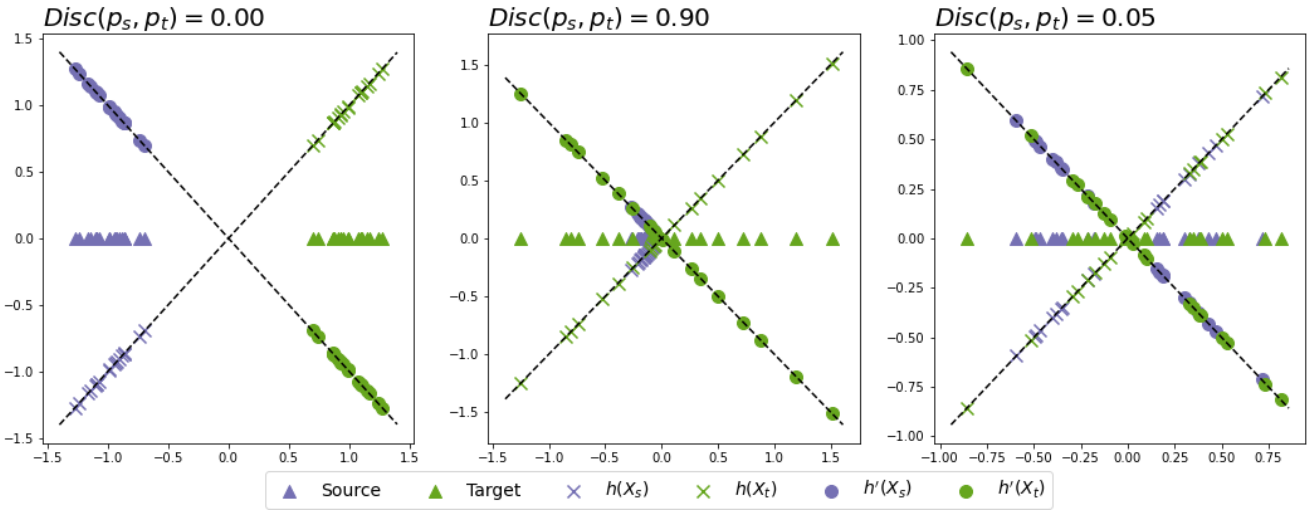
Zhao et al. (2018) [196] directly used the DANN presented above for regression by simply changing the prediction loss $\mathcal{L}_y$ to the ℓ^2 loss, and obtain good experimental results. We argue that this is in general ill-advised as the $d_{\mathcal{H}\Delta d_{\mathcal{H}}}$ is in general uninformative for the actual discrepancy between domains induced by regressors.

We highlight our argument with a simple example. We simply generate three source and target datasets using 1D-Gaussians, ie $X_S \sim \mathcal{N}(\mu_S, \sigma_S)$ and $X_T \sim \mathcal{N}(\mu_T, \sigma_T)$. We chose in the first example $\mu_S = 1$, $\mu_T = 1$ and small values of $\sigma_S = \sigma_T << 1$, in the second example, $\mu_S = 1$, $\mu_T = 1$ and $\sigma_S << \sigma_T$, finally $\mu_S = \mu_T = 0$ and $\sigma_S = 1$ and $\sigma_T = 4$in the last example.

[1] In [72], authors use the un-corrected theorem of [24] with $d_{\mathcal{H}}$. Their motivation still holds as Ben-David et al. (2010) [25] already proposed to approximate $d_{\mathcal{H}}$ with $d_{\mathcal{H}\Delta\mathcal{H}}$.



(a) $d_{\mathcal{H}}$ with one-dimensional data



(b) Discrepancy with one-dimensional data

Figure 4.2: $\mathcal{H}$-Divergence vs Discrepancy for 1-Dimensional Data

On one hand, Figure 4.2a shows that in the first scenario, using a linear classifier, the domains are perfectly separated, in the last scenario, they are completely mixed and in the second scenario, they are semi-separated. On the other hand, when computing the discrepancy using bounded regressors (*ie* $\mathcal{H} = \{h : x \to w^T x; \|w\|_2 \leq 1\}$), in the first case the domains are perfectly aligned. In the semi-separated case, domains are considered very far by the Discrepancy. This simple example shows the hardness of domain adaptation for regression, as the output space is continuous. Hence, to be able to perform domain adversarial adaptation, we introduce a general theory that encompasses both classification and regression.

4.2 Hypothesis-Discrepancy for Domain Adaptation in Regression

4.2.1 Hypothesis-Discrepancy

The concept of discrepancy [121], defined in Section 2.2, was introduced to estimate the shift between domains for more general problems. We recall its definition:

Definition 9. *For two probability measures P and Q defined over a set $\mathcal{X}$, an hypothesis class $\mathcal{H} : \mathcal{X} \to \mathcal{Y}$ and a*

loss $L :: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$, the discrepancy between P and Q is defined as

$$\textit{disc}_{\mathcal{H},L}(P,Q) = \sup_{h,h' \in \mathcal{H}} \left| \mathop{\mathbb{E}}_{x \sim P}[L(h(x),h'(x))] - \mathop{\mathbb{E}}_{x \sim Q}[L(h(x),h'(x))] \right| \tag{4.2}$$

One of the main difficulties when estimating the discrepancy is due to the computation of the supremum over two hypotheses h and h'. For this reason, we introduce the concept of hypothesis-discrepancy as follows

Definition 10. *For two distributions P, Q over a set $\mathcal{X}$ and for a hypothesis class $\mathcal{H}$ over $\mathcal{X}$, for any $h \in \mathcal{H}$, the hypothesis-discrepancy (or $HDisc$) associated with h is defined as:*

$$HDisc_{\mathcal{H},L}(P,Q;h) = \sup_{h' \in \mathcal{H}} \left| \mathop{\mathbb{E}}_{x \sim P}[L(h(x),h'(x))] - \mathop{\mathbb{E}}_{x \sim Q}[L(h(x),h'(x))] \right| \tag{4.3}$$

For any given $h \in \mathcal{H}$ hypothesis-discrepancy measures a similarity between two distributions. It is directly dependent on the hypothesis class $\mathcal{H}$ and the loss L. In the definition, h' can be seen as a predictor that would be very close to h on the source domain but far on the target domain (or vice-versa). Theorem 10 states that hypothesis-discrepancy can be estimated with finite samples.

Theorem 10. *We assume that the loss L is symmetric, follows the triangle inequality and there exists $M > 0$ such that L verifies $L(h(x),y) \leq M$ for all $h \in \mathcal{H}$ and $(x,y) \in \mathcal{X} \times \mathcal{Y}$. Then for two distributions P_S and P_T defined over $\mathcal{X}$, for any hypothesis $h \in \mathcal{H}$, with probability $1-\delta$ over the samples of $\mathcal{S}_S$ of size m according to P_S and $\mathcal{S}_T$ of size n according to P_T the following bound holds:*

$$HDisc_{\mathcal{H},L}(P_S,P_T;h) \leq HDisc_{\mathcal{H},L}(\hat{P_S},\hat{P_T};h) + 2\mathcal{R}_m(\mathcal{H}_s) + 2\mathcal{R}_n(\mathcal{H}_t) + M\sqrt{\frac{\log 2/\delta}{2m}} + M\sqrt{\frac{\log 2/\delta}{2n}} \tag{4.4}$$

where $\hat{P_S}$ and $\hat{P_T}$ are the empirical estimates of P_S and P_T induced by $\mathcal{S}_s$ and $\mathcal{S}_s$. $\mathcal{R}_m(\mathcal{H}_s)$ and $\mathcal{R}_n(\mathcal{H}_t)$ are the Rademacher complexities of $\mathcal{H}_s$ and $\mathcal{H}_t$ defined in Definition 3.

Proof. Let us consider the empirical distribution $\hat{P_S}$ corresponding to a sample $\mathcal{S}_s$ of size m and a hypothesis $h \in \mathcal{H}$. We first define $\phi(h) = HDisc(P_S,P_T;h) - HDisc(\hat{P_S},P_T;h)$. Then as the loss L is bounded by M, changing one element of $\mathcal{S}_s$ changes $\phi(h)$ by a maximum of $\frac{M}{m}$. McDiarmid's inequality [127] states that with probability $1-\frac{\delta}{2}$,

$$HDisc(P_S,P_T;h) \leq HDisc(\hat{P_S},P_T;h) + \mathbb{E}_{\widehat{P_S}}[\phi(h)] + M\sqrt{\frac{\log(2/\delta)}{2m}} \tag{4.5}$$

Moreover, from Theorem 2 and Proposition 2 from [121], we know that

$$\mathbb{E}_{\widehat{P_S}}[\phi(h)] = \mathbb{E}_{\widehat{P_S}}[HDisc(\hat{p_s},p_t;h) - HDisc(\hat{p_s},p_t;h)] \leq 2\mathcal{R}_m(\mathcal{H}_s)$$

where $\mathcal{H}_s = \{x \to L(h(x), f_S(x)); \forall h \in \mathcal{H}\}$. As a consequence, with probability $1 - \frac{\delta}{2}$ over the sampling of $\mathcal{S}_s$:

$$HDisc(P_S, P_T; h) \leq HDisc(\hat{P_S}, P_T; h) + 2\mathcal{R}_m(\mathcal{H}_s) + M\sqrt{\frac{\log(2/\delta)}{2m}} \tag{4.6}$$

Using the same reasoning on P_T, we obtain that with probability $1 - \frac{\delta}{2}$ over the sampling of $\mathcal{S}_t$:

$$HDisc_{\mathcal{H},L}(P_S, P_T; h) \leq HDisc_{\mathcal{H},L}(\hat{P_S}, \hat{P_T}; h) + 2\mathcal{R}_m(\mathcal{H}_s) + 2\mathcal{R}_n(\mathcal{H}_t) + M\sqrt{\frac{\log 2/\delta}{2m}} + M\sqrt{\frac{\log 2/\delta}{2n}} \tag{4.7}$$

Hence using an union bound, we have the final result with probability $1 - \delta$:

$$HDisc_{\mathcal{H},L}(P_S, P_T; h) \leq HDisc_{\mathcal{H},L}(\hat{P_S}, \hat{P_T}; h) + 2\mathcal{R}_m(\mathcal{H}_s) + 2\mathcal{R}_n(\mathcal{H}_t) + M\sqrt{\frac{\log 2/\delta}{2m}} + M\sqrt{\frac{\log 2/\delta}{2n}} \tag{4.8}$$

□

Hypothesis-Discrepancy is directly linked to the original discrepancy by $\text{Disc}_{\mathcal{H},L}(P_S, P_T) = \sup_{h \in \mathcal{H}} HDisc_{\mathcal{H},L}(P_S, P_T; h)$. Kuroki et al. (2019) [104] also have noted that the presence of a supremum over two hypotheses is suboptimal and introduced the source-discrepancy which is a specific case of our hypothesis-discrepancy with $h = h_s^*$.

4.2.2 Domain Adaptation Guarantees with Hypothesis-Discrepancy

Using $HDisc$, we are able to show the following theorem for unsupervised single source domain adaptation:

Theorem 11. *If L is symmetric and follows the triangle inequality, then the following bound holds:*

$$\mathbf{R_T}(h, f_T) \leq \mathbf{R_S}(h, f_S) + \eta_{\mathcal{H}}(f_S, f_T) + HDisc_{\mathcal{H},L}(p_t, p_s; h) \tag{4.9}$$

where

$$\eta_{\mathcal{H}}(f_S, f_T) = \min_{h_0 \in \mathcal{H}} \mathbf{R_T}(h_0, f_T) + \mathbf{R_S}(h_0, f_S)$$

Proof. For any $h \in \mathcal{H}$,

$$
\begin{aligned}
\mathbf{R_T}(h, f_T) &\leq \mathbf{R_S}(h, f_S) + |\mathbf{R_T}(h, f_T) - \mathbf{R_S}(h, f_S)| \\
&\leq \mathbf{R_S}(h, f_S) + |\mathbf{R_T}(h, h_0) - \mathbf{R_T}(h, f_T)| + |\mathbf{R_S}(h, h_0) - \mathbf{R_S}(h, f_S)| + |\mathbf{R_T}(h, h_0) - \mathbf{R_S}(h, h_0)| \\
&\leq \mathbf{R_S}(h, f_S) + HDisc_{\mathcal{H},L}(p_s, p_t; h) + \mathbb{E}_{x\sim p_t}[|L(h(x), h_0(x)) - L(h(x), f_T(x))|] \\
&+ \mathbb{E}_{x\sim p_s}[|L(h(x), h_0(x)) - L(h(x), f_T(x))|]] \\
&\leq \mathbf{R_S}(h, f_S) + HDisc_{\mathcal{H},L}(p_s, p_t; h) + \mathbb{E}_{x\sim p_t}[|L(h_0(x), f_T(x))|] + \mathbb{E}_{x\sim p_s}[|L(h_0(x), f_S(x))|] \\
&\leq \mathbf{R_S}(h, f_S) + \mathbf{R_S}(h_0, f_S) + \mathbf{R_T}(h_0, f_T) + HDisc_{\mathcal{H},L}(p_s, p_t; h)
\end{aligned}
$$

where the first inequality comes from the triangle inequality, the second holds for any $h_0 \in \mathcal{H}$. The third comes from the definition of $HDisc(p_s, p_t)$ and the fourth follows from triangle inequality on L. Finally, the result follows taking the minimum over all h_0. □

The term $\mathbf{R_S}(h, f_S)$ characterizes the source risk and can be controlled. The second term measures the joint error of an ideal hypothesis on both domains. As it involves f_T it cannot be controlled without further assumptions in unsupervised domain adaptation but one can still notice it is smaller than the one of Equation 2.19. We assume it to be small *ie* $f_S \simeq f_T$.

The last term is dependent on h and involves only one supremum over $h' \in \mathcal{H}$. Hence while our hypothesis-discrepancy is less general than the discrepancy it allows tighter bounds for Domain Adaptation. Using Theorem 10, one can see that $HDisc$ can be estimated with finite samples, as can $\mathbb{R}_S(h, f_s)$ using generalization bounds seen in Section 2.2.1.

4.3 Minimizing the hypothesis-discrepancy

Thanks to hypothesis-discrepancy, we proved that the target risk can be controlled by the sum of the source risk and the hypothesis-discrepancy under the assumption that η is small. The source risk can be estimated and minimized using a traditional learning algorithm. However the hypothesis-discrepancy requires a supremum and is dependent on the predictor h.

We propose to formulate an objective corresponding to our bound with a min-max problem. Hence we formulate the following objective for Adversarial Hypothesis Discrepancy Minimization (**AHDM**):

$$
\min_{h\in\mathcal{H}} \max_{h'\in\mathcal{H}} \mathbf{R_S}(h, f_S) + |\mathbf{R_S}(h, h') - \mathbf{R_T}(h, h')| \tag{4.10}
$$

Equation 4.1 and DANN share a similar objective. Hence, we now propose a general adversarial domain adaptation algorithm to learn transferable representations. We introduce a feature extractor $\phi_\theta : \mathcal{X} \to \mathcal{Z}$ parametrized by

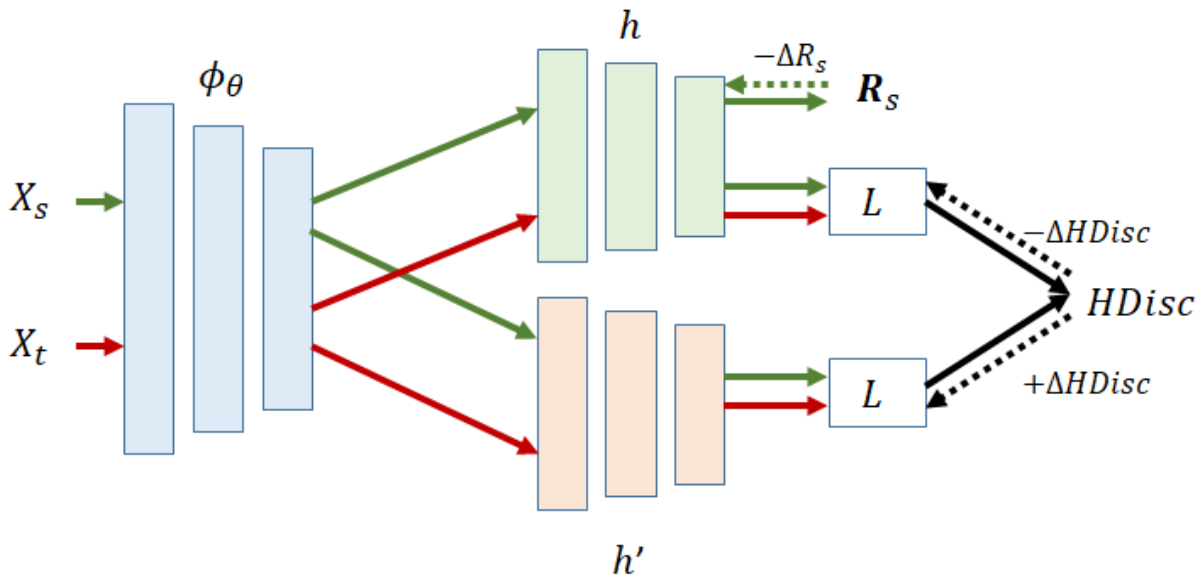


Figure 4.3: Adversarial Hypothesis Discrepancy Minimization (**AHDM)** using Neural Networks

parameters θ and a class of predictor $\mathcal{H}_\mathcal{Z} : \mathcal{Z} \to \mathcal{Y}$. Given unlabeled target sample $\mathcal{S}_T = \{(x_T^{(1)}, ..., x_T^{(n_T)}\} \in \mathbb{R}^{n_T \times d}$ and a labeled source sample $\mathcal{S}_S = \{(x_S^{(1)}, ..., x_S^{(n_S)}\} \in \mathbb{R}^{n_S \times d}$ with labels $\{y_S^{(1)}, ..., y_S^{(n_S)}\} \in \mathbb{R}^{n_S}$, we minimize the combination of the source risk and the hypothesis-discrepancy between the marginal weighted source distribution and target distribution. We formulate the following objective for Adversarial Discrepancy Minimization (**ADM**):

$$\min_{\phi_\theta, h \in \mathcal{H}_\mathcal{Z}} \max_{h' \in \mathcal{H}_\mathcal{Z}} \mathbf{R_S}(h \circ \phi_\theta, y_S) + |\mathbf{R_T}(h \circ \phi_\theta, h' \circ \phi_\theta) - \mathbf{R_S}(h \circ \phi_\theta, h' \circ \phi_\theta)| \tag{4.11}$$

In order to minimize this objective, we use a neural network as described in Figure 4.3. In practice we consider h and h' to be neural networks with constrained weights. The parameters of each part of the network are minimized successively by:

1. $\mathcal{L}_h = \mathbf{R_S}$ updates h to minimize the source loss
2. $\mathcal{L}_{h'} = -HDisc$ updates h' to maximize discrepancy
3. $\mathcal{L}_\theta = HDisc + \mathbf{R_S}$ updates ϕ_θ to minimize discrepancy and source loss

We use the traditional feed-forward back-propagation to minimize those losses. To the best of our knowledge, this is the first adversarial method proposed working in the regression case.

Classifiers When working on classification, a link can be made between our ADM and the Maximum Classifier Discrepancy (MCD). In MCD, the loss minimized by h and h' both are $\mathbf{R_S} - \mathbf{R_T}(h, h')$. Under the assumption that h and h' are very similar on $\mathcal{D}_S$, $\mathbf{R_S}(h, h') \simeq 0$. Hence, under this assumption, $|\mathbf{R_T}(h, h') - \mathbf{R_S}(h, h')| \simeq \mathbf{R_T}(h, h')$, which is exactly the ”discrepancy” used by the MCD method. Hence, MCD can be seen as a specific case of our

method where (i) the hypothesis space $\mathcal{H}$ is restricted to "good" classifiers on the source data (ii) the problem is a classification problem.

Linear Regressors Another interesting case is the case of ℓ^2-loss with constrained linear regressors, *ie* $\mathcal{H}_\mathcal{Z} = \{h : x \to w^T x; \|w\|_2 \leq 1\}$. In this case, we can develop a closed-form estimation of *HDisc*. Indeed, let us write $h(x) = w^T x$ and $h'(x) = w'^T x$. If, for two samples $\mathbf{X_S} = [x_S^{(1)}, ..., x_S^{(n_S)}] \in \mathbb{R}^{n_S \times d}$ and $\mathbf{X_T} = [x_T^{(1)}, ..., x_T^{(n_T)}] \in \mathbb{R}^{n_T \times d}$ we denote $\mathbf{Z_S} = \phi_\theta(\mathbf{X_S})$ and $\mathbf{Z_T} = \phi_\theta(\mathbf{X_T})$ the output of the feature extractor for source and target data, then

$$\begin{aligned} \textit{HDisc}_{\mathcal{H}_\mathcal{Z}, \ell^2}(\mathbf{Z_S}, \mathbf{Z_T}; h) &= \sup_{||w'||_2 \leq 1} \left| \frac{1}{n}(w - w')^T \mathbf{Z_S}^T \mathbf{Z_S}(w - w') - \frac{1}{m}(w - w')^T \mathbf{Z_T}^T \mathbf{Z_T}(w - w') \right| \\ &\leq \sup_{||u||_2 \leq 2} \left| u^T \left(\frac{1}{n} \mathbf{Z_S}^T \mathbf{Z_S} - \frac{1}{m} \mathbf{Z_T}^T \mathbf{Z_S} \right) u \right| \\ &= 2 \left\| \frac{1}{n} \mathbf{Z_S}^T \mathbf{Z_S} - \frac{1}{m} \mathbf{Z_T}^T \mathbf{Z_T} \right\|_2 \end{aligned} \tag{4.12}$$

where $\|.\|_2$ denotes the spectral norm. The spectral norm can be computed using SVD or power method, and the gradients can be computed as well using the eigendecomposition and feeded to the feature extractor ϕ_θ. Adlam et al. (2019) [2] used a similar idea to train GANs where they show that minimizing the discrepancy comes down to bringing closer the covariance of the extracted features. This is similar to the CORAL [170] and its extension Deep CORAL [169] where a similar regularization on the covariance matrix is used. As such, our framework provides a theoretical ground to this method and shows that it should work particularly well with regression.

4.4 Extension to multiple sources

When multiple sources are available, a straightforward idea is to merge all the source domains into one and transform the problem to a single-source domain adaptation where Theorem 11 holds. This solution is clearly not optimal as different source domains may have different relationships to the target one.

The framework of Unsupervised Multiple Source Domain Adaptation is defined as follows: we consider K independent source domains $\mathcal{D}_k = \{\mathcal{X}, P_k\}$ with associated labelling function f_k and a target domain $\mathcal{D}_T = \{\mathcal{X}, P_T\}$ with associated labelling function f_T. Once again, the goal is to minimize the target risk $\mathbf{R_T}(h) = \mathbb{E}_{x \sim P_T}[L(h(x), f_T(x))]$. We introduce a new bound relating the target risk with a weighted combination of the source risks $\mathbf{R_k}(h)$.

4.4.1 Theoretical Guarantees with multiple sources

We propose to attribute weights to each source and introduce the α-weighted source domain $\mathcal{D}_\alpha = \{p_\alpha, f_\alpha\}$ such that for $\alpha \in \Delta = \{\alpha \in \mathbb{R}^K; \alpha_k \geq 0, \sum_{k=1}^K \alpha_k = 1\}$, $f_\alpha : x \to (\sum_{k=1}^K \alpha_k p_k(x) f_k(x))/(\sum_{j=1}^K \alpha_j p_j(x))$ and $p_\alpha = \sum_{k=1}^K \alpha_k p_k$.

In this framework, the objective is that sources that are unrelated to the target are given low weights whereas sources closely related to the target have a weight close to 1. The α-weighted sample is defined as $\mathcal{S}_\alpha = \bigcup_{k=1}^{K} \mathcal{S}_k$ with probabilities $\hat{p_\alpha}(x_k^{(i)}) = \alpha_k/m$. Similarly, we consider an unlabeled target sample $\mathcal{S}_t = \{(x_1^{(t)}, ..., x_n^{(t)})\}$ where $x_T^{(i)} \overset{i.i.d}{\sim} p_t$ and define the sets $\mathcal{H}_k = \{g : x \to L(h(x), f_k(x)); h \in \mathcal{H}\}$.

Theorem 12. *Assuming that the loss L is symmetric and follows the triangle inequality, then for any hypothesis $h \in \mathcal{H}$, with probability $1 - \delta$ the following bound holds:*

$$\begin{aligned}\mathbf{R_T}(h, f_T) \leq &\sum_{k=1}^{K} \alpha_k \hat{\mathbf{R}}_\mathbf{k}(h, f_k) + HDisc_{\mathcal{H},L}(p_t, p_\alpha) + \eta_{\mathcal{H},\alpha} \\ &+ 2\sum_{k=1}^{K} \alpha_k \mathcal{R}_m(\mathcal{H}_k) + \|\alpha\|_2 M\sqrt{\frac{\log(1/\delta)}{2m}}\end{aligned} \tag{4.13}$$

where

- $\eta_{\mathcal{H},\alpha} = \min_{h_0 \in \mathcal{H}} [\mathbf{R}_\alpha(h_0, f_\alpha) + \mathbf{R_T}(h_0, f_T)]$
- $\mathcal{R}_m(\mathcal{H}_k)$ *is the Rademacher complexity of* $\mathcal{H}_k = \{h : x \to L(h(x), f_k(x)); h \in \mathcal{H}\}$

Proof. Using Theorem 11, with p_α, we get:

$$\mathbf{R_T}(h, f_T) \leq \mathbf{R}_\alpha(h, f_\alpha) + \eta_\mathcal{H}(f_T, f_\alpha) + HDisc_{\mathcal{H},L}(p_\alpha, p_t; h) \tag{4.14}$$

Let us consider K empirical distributions $\hat{p_k}$ corresponding to a sample $\mathcal{S}_k = \{x_1^{(k)}, ..., x_m^{(k)}\}$ of size m and a hypothesis $h \in \mathcal{H}$. Then the α-weighted sample $\mathcal{S}_\alpha$ is of size Km and empirical distribution $\hat{p_\alpha}$ defined as $\hat{p_\alpha}(x_k^{(i)}) = \frac{\alpha_k}{m}$. We define $\phi = \mathbf{R}_\alpha(h, f_\alpha) - \hat{\mathbf{R}}_\alpha(h, f_\alpha)$. Noting that changing an element $x_k^{(i)}$ modifies ϕ by a maximum of $\frac{\alpha_k}{m}$, we can deduce from the McDiarmid's inequality that:

$$\mathbb{P}(\phi - \mathbb{E}(\phi) \leq t) \leq \exp^{\frac{-2mt^2}{\sum_{k=1}^{K} \alpha_k^2}} \tag{4.15}$$

Hence with probability $1 - \delta$,

$$\mathbf{R}_\alpha(h, f_\alpha) \leq \hat{\mathbf{R}}_\alpha(h, f_\alpha) + \mathbb{E}_{\hat{p_\alpha}}[\phi] + \|\alpha\|_2 M\sqrt{\frac{\log 1/\delta}{2m}} \tag{4.16}$$

We are able to compute $\mathbb{E}_{\hat{p_\alpha}}$ using the classical tool of ghost sample in Rademacher complexity analysis. In the following the random variable σ takes its values uniformly over $\{-1, 1\}$. We denote $\{x_k^{(i)}; 1 \leq k \leq K, 1 \leq i \leq m\}$ the sample associated to $\hat{p}_\alpha \sim p_\alpha$ and $\{z_i^{(k)}; 1 \leq k \leq K, 1 \leq i \leq m\}$ $\hat{q}_\alpha \sim p_\alpha$.

$$
\begin{aligned}
\mathbb{E}_{\hat{p_\alpha}\sim p_\alpha}[\phi] &= \mathop{\mathbb{E}}_{\hat{p_\alpha}}\left[\mathbf{R}_\alpha(h,f_\alpha) - \hat{\mathbf{R}}_\alpha(h,f_\alpha)\right] \\
&\leq \mathop{\mathbb{E}}_{\hat{p_\alpha}\sim p_\alpha}\left[\sup_{h\in\mathcal{H}} \mathbf{R}_\alpha(h,f_\alpha) - \hat{\mathbf{R}}_\alpha(h,f_\alpha)\right] \\
&\leq \mathop{\mathbb{E}}_{\hat{p_\alpha}\sim p_\alpha}\left[\sup_{h\in\mathcal{H}} \mathbb{E}_{\hat{q_\alpha}\sim p_\alpha}[\mathbb{E}_{x\sim\hat{q_\alpha}}[L(h(x),f_\alpha(x))]] - \mathbb{E}_{x\sim\hat{p_\alpha}}[L(h(x),f_\alpha(x))\right] \\
&\leq \mathop{\mathbb{E}}_{\hat{p_\alpha},\hat{q_\alpha}\sim p_\alpha}\left[\sup_{h\in\mathcal{H}} \mathbb{E}_{x\sim\hat{q_\alpha}}[L(h(x),f_\alpha(x))] - \mathbb{E}_{x\sim\hat{p_\alpha}}[L(h(x),f_\alpha(x))\right] \\
&\leq \mathop{\mathbb{E}}_{\hat{p_\alpha},\hat{q_\alpha}\sim p_\alpha}\left[\sup_{h\in\mathcal{H}} \sum_{k=1}^{K}\alpha_k(\mathbb{E}_{x\sim\hat{q_k}}[L(h(x),f_k(x)) - \mathbb{E}_{x\sim\hat{p_k}}[L(h(x),f_k(x))]\right] \\
&\leq \mathop{\mathbb{E}}_{\hat{p_\alpha},\hat{q_\alpha}\sim p_\alpha}\left[\sup_{h\in\mathcal{H}} \sum_{k=1}^{K}\frac{\alpha_k}{m}\sum_{i=1}^{m}(L(h(x_k^{(i)}),f_k(x_k^{(i)})) - L(h(z_i^{(k)}),f_k(z_i^{(k)})))\right] \\
&\leq \mathop{\mathbb{E}}_{\substack{\hat{p_\alpha},\hat{q_\alpha}\sim p_\alpha \\ \sigma_i\sim\sigma}}\left[\sup_{h\in\mathcal{H}} \sum_{k=1}^{K}\frac{\alpha_k}{m}\sum_{i=1}^{m}\sigma_i(L(h(x_k^{(i)}),f_k(x_k^{(i)})) - L(h(z_i^{(k)}),f_k(z_i^{(k)})))\right] \\
&\leq \mathop{\mathbb{E}}_{\substack{\hat{p_\alpha}\sim p_\alpha \\ \sigma_i\sim\sigma}}\left[\sup_{h\in\mathcal{H}} \sum_{k=1}^{K}\frac{\alpha_k}{m}\sum_{i=1}^{m}\sigma_i L(h(x_k^{(i)}),f_k(x_k^{(i)}))\right] \\
&\quad + \mathop{\mathbb{E}}_{\substack{\hat{q_\alpha}\sim p_\alpha \\ \sigma_i\sim\sigma}}\left[\sup_{h\in\mathcal{H}} -\sum_{k=1}^{K}\frac{\alpha_k}{m}\sum_{i=1}^{m}\sigma_i L(h(z_i^{(k)}),f_k(z_i^{(k)}))\right] \\
&\leq 2\mathop{\mathbb{E}}_{\substack{\hat{p_\alpha}\sim p_\alpha \\ \sigma_i\sim\sigma}}\left[\sup_{h\in\mathcal{H}} \sum_{k=1}^{K}\frac{\alpha_k}{m}\sum_{i=1}^{m}\sigma_i L(h(x_k^{(i)}),f_k(x_k^{(i)}))\right] \\
&\leq 2\sum_{k=1}^{K}\alpha_k\mathcal{R}_m(\mathcal{H}_k)
\end{aligned}
\tag{4.17}
$$

Finally, noting that with empirical distributions $\hat{\mathbf{R}}_\alpha(h,f_\alpha) = \sum_{k=1}^{K}\alpha_k\hat{\epsilon_k}(h,f_k)$, we obtain the final result with probability $1-\delta$ over the sample of $\mathcal{S}_1$, ..., $\mathcal{S}_K$:

$$
\begin{aligned}
\mathbf{R_T}(h,f_T) \leq &\sum_{k=1}^{K}\alpha_k\hat{\mathbf{R}}_\mathbf{k}(h,f_k) + HDisc_{\mathcal{H},L}(p_t,p_\alpha) + \eta_{\mathcal{H},\alpha} \\
&+ 2\sum_{k=1}^{K}\alpha_k\mathcal{R}_m(\mathcal{H}_k) + \|\alpha\|_2 M\sqrt{\frac{\log(1/\delta)}{2m}}
\end{aligned}
\tag{4.18}
$$

□

Theorem 12 presents a theoretical analysis in the multi-source domain adaptation framework. The first term corresponds to the α-weighted source risks and can be controlled by learning h close to f_k. The second term connects the target distribution with the α-weighted source distribution. The third term is related to how different the labelling functions on the target and the source are and is expected to be small in unsupervised domain adaptation. The last two terms are related to the convergence rate of this bound and Cortes et al. (2017) [45] proved that

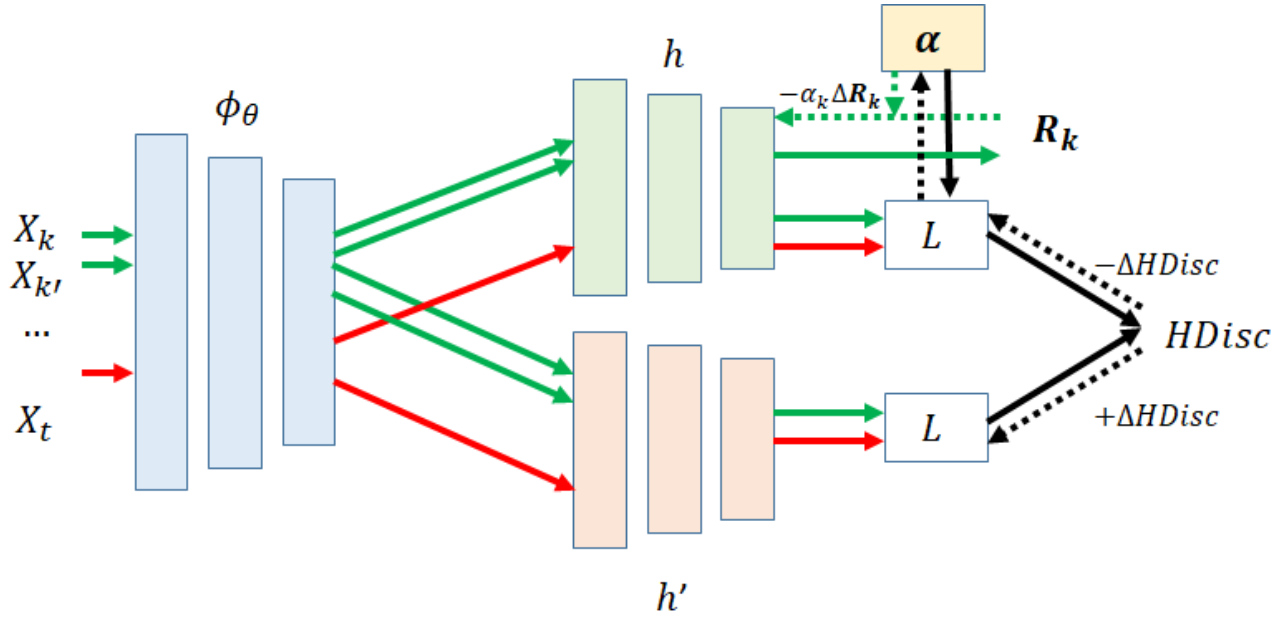


Figure 4.4: Adversarial Multi-Source Hypothesis Discrepancy Minimization (**AMSHDM)** The adversarial scheme is similar to single-source with weights α. At each iteration, the weights α are updated.

$\mathcal{R}_m(\mathcal{H}_k) = \mathcal{O}(1/\sqrt{m})$ for neural networks.

In order to adapt from the K sources to the target, we need to minimize the hypothesis-discrepancy between the α-weighted domain and the target domain. We propose in the next section an algorithm to find ideal representations of the sources and weights to select the best sources for adaptation.

4.4.2 Algorithm

We now present a practical solution for the case of multiple sources. We introduce a feature extractor parametrized by θ $\phi_\theta : \mathcal{X} \to \mathcal{Z}$ and a class of predictor $\mathcal{H}_\mathcal{Z} : \mathcal{Z} \to \mathcal{Y}$. Given unlabeled target sample $\mathcal{S}_T = \{x_T^{(1)}, ..., x_T^{(n_T)}\} \in \mathbb{R}^{n_T \times d}$ and K labeled sources $\mathcal{S}_k = \{x_k^{(1)}, ..., x_k^{(m)}\} \in \mathbb{R}^{m \times d}$ with labels $Y_k = \{y_k^{(1)}, ..., y_k^{(m)}\} \in \mathbb{R}^m$, our aim is to minimize the combination of the source risk and the hypothesis-discrepancy between the marginal weighted source distribution and target distribution as in Theorem 12.

Using the definition of $HDisc$, we formulate the following objective for our Adversarial Multi-Source Hypothesis-Discrepancy Minimization (**AMSHDM**), illustrated in Figure 4.4:

$$\min_{\substack{\phi_\theta, h \in \mathcal{H} \\ \|\alpha\|_1 = 1}} \max_{h' \in \mathcal{H}} \Bigg[\sum_{k=1}^{K} \alpha_k \mathbf{R_k}(h \circ \phi_\theta, y_k) + \lambda \|\alpha\|_2 + \left| \mathbf{R_T}(h \circ \phi_\theta, h' \circ \phi_\theta) - \sum_{k=1}^{K} \alpha_k \mathbf{R_k}(h \circ \phi_\theta, h' \circ \phi_\theta) \right| \Bigg] \tag{4.19}$$

Algorithm 1 Pseudo-algorithm for **AMSHDM**

Initialize $\alpha_k = \frac{1}{K}$, h, h' and θ randomly, choose learning rates η_h, η_θ and η_α
for $e = 1...epochs$ **do**
Forward propagation
$\mathbf{R_k} = \frac{1}{m}\sum_{i=1}^{m} L(h^{(e)}(\phi_{\theta^{(e)}}(x_k^{(i)}), y_i^{(k)}))$
$HDisc = \left|\mathbf{R_T}(h^{(e)} \circ \phi_{\theta^{(e)}}, h'^{(e)} \circ \phi_{\theta^{(e)}}) - \sum_{k=1}^{K} \alpha_k \mathbf{R_k}(h^{(e)} \circ \phi_{\theta^{(e)}}, h'^{(e)} \circ \phi_{\theta^{(e)}})\right|$
Backward propagation
$h^{(e+1)} \leftarrow h^{(e)} - \eta_h \left(\sum_{k=1}^{K} \alpha_k^{(e)} \Delta_h \mathbf{R_k}(h^{(e)})\right)$ ▷ $(*)$
$h'^{(e+1)} \leftarrow h'^{(e)} + \eta_h \left(\sum_{k=1}^{K} \alpha_k^{(e)} \Delta_{h'} HDisc(h'^{(e)})\right)$
$\theta^{(e+1)} \leftarrow \theta^{(e)} - \eta_\theta \left(\sum_{k=1}^{K} \alpha_k^{(e)} \Delta_\theta \mathbf{R_k}(\theta^{(e)}) + \Delta_\theta HDisc(\theta^{(e)})\right)$
$\alpha_k^{(e+1)} \leftarrow \alpha_k^{(e)} - \eta_\alpha \left(\Delta_{\alpha_k} HDisc(\alpha_k^{(e)}) + 2\lambda\alpha_k^{(e)}\right)$
Clip weights of $\phi_\theta^{(e+1)}$, $h^{(e+1)}$ and $h'^{(e+1)}$
$\alpha^{(e+1)} = \alpha^{(e+1)} / \|\alpha^{(e+1)}\|_1$
end for
$(*)$ For a parameter p and a loss L, we note $\Delta_p L(p_0)$ the gradient of L with respect to p computed at p_0

where λ is a hyperparameter. The first term of the objective leads h to be a good predictor on the source task. For any given h and h' the discrepancy term constrains both representations ϕ_θ and weights α to align domains. The term $\eta_\mathcal{H}$ is ignored in our objective and assumed to be small.

Similarly to the single-source scenario, we sequentially optimize different parameters of our networks according to different objectives. At a given iteration, four losses are minimized sequentially:

1. $\mathcal{L}_h = \alpha_k \mathbf{R_k}$ updates h to minimize the source loss
2. $\mathcal{L}_{h'} = -HDisc$ updates h' to maximize discrepancy
3. $\mathcal{L}_\theta = HDisc + \sum_{k=1}^{K} \alpha_k \mathbf{R_k}$ updates ϕ_θ to minimize discrepancy and source loss
4. $\mathcal{L}_\alpha = HDisc + \lambda||\alpha||_2$ updates α to minimize the discrepancy

The loss $\mathcal{L}_\alpha$ only contains the discrepancy term. Indeed, our goal is to select the domains closer to the source in terms of discrepancy. Including the source loss in $\mathcal{L}_\alpha$ may give too large weights to sources that are "easy" to predict. It is possible to keep the term with a μ parameter to control its influence but in our experiments, it did not bring any improvement. It is also be possible to completely update α at each epoch but it appears to be sub-performing. Our method allows the weights to smoothly adapt to the representations learnt by ϕ_θ. The weighting scheme is generic and can also be used with the specific cases of Discrepancy mentioned above (linear regressors for instance).

4.5 Experiments

We conducted extensive experiments for our method, both on synthetic and real data. Experiments on publicly available data are available on our git repository (https://github.com/GRichard513/ADisc-MSDA). For applications on NILM, we give details to make the experiments reproducible. More details about architectures can also be found in

Appendix C.4.

4.5.1 Synthetic data

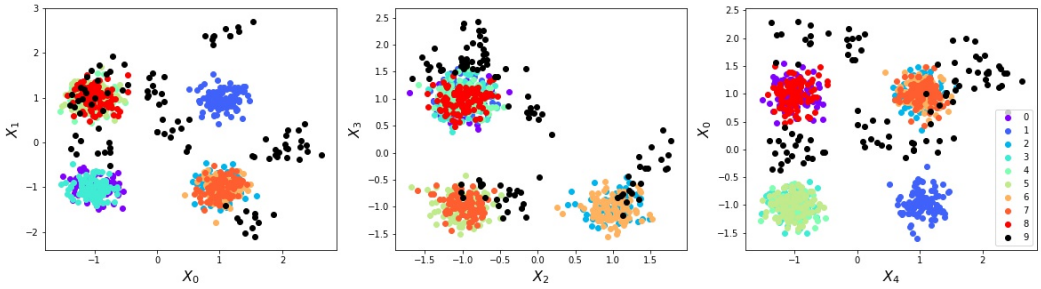


Figure 4.5: Data for the single-source Friedman experiment over the 5 features with $\sigma_k = 0.2$, $\sigma_c = 0.2$, $\mu_{shift} = 0.5$, $\sigma_{shift} = 0.5$. From right to left: (x_0, x_1) ; (x_2, x_3); $(x_4; x_0)$. Each color corresponds to a source, the target is in black.

We first run experiments for **AHDM** and **AMSHDM** on synthetic data to study the behaviour of our algorithm. To generate synthetic data, we use a modified version of the Friedman regression problem [70] consisting of inputs x of dimension 5 and a prediction function defined as

$$y(x) = 10sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \epsilon$$

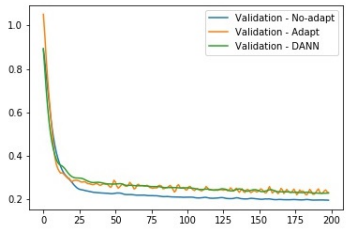
where $\epsilon \sim \mathcal{N}(0, \sigma_y)$.

To highlight the two contributions of our method, our goal is two-fold: we firstly aim to demonstrate the effectiveness of $HDisc$ in the single-source DA scenario (K=1 and $\alpha = 1$). Then we run different experiments to show when multi-source DA is expected to bring an improvement.

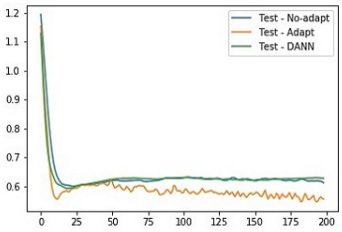
We generate source data as follows: we define three clusters of sources $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3\}$, $\{\mathcal{S}_4, \mathcal{S}_5, \mathcal{S}_6\}$ and $\{\mathcal{S}_7, \mathcal{S}_8, \mathcal{S}_9\}$. For each cluster, and for each feature $(1...5)$, a mean of -1 or 1 is selected at random. Then each mean of each source of a given cluster is shifted by a random shift i.e. for a source k associated to a cluster c $\mu_k^{(f)} \sim \mathcal{N}(\mu_c, \sigma_c)$. Finally, each source sample is randomly chosen with a normal distribution $p_k = \mathcal{N}(\mu_k^{(f)}, \sigma_k)$. In Figure 4.5, we display each feature of the generated data.

In order to separate the effects of our two contributions, we split the experiments in two parts: in the first experiment, demonstrate the effectiveness of our hypothesis-discrepancy minimization in the single source scenario (i.e. when $\alpha_k = \frac{1}{K}$) and why the classical $d_{\mathcal{H}}$ fails in this regression scenario. In a second scenario, we show how our weighting scheme helps adaptation in multi-source, especially to select sources that are related to the target.

Single-Source DA For single-source, we merge all 9 source domains together to create one. The target sample is generated choosing uniformly one of the source domains for each element and adding a noise of the form $\mathcal{N}(\mu_{shift},$



(a) Validation loss on source data



(b) Target loss

Figure 4.6: Training curves for Single Source DA: without adaptation, the target loss increases as the validation loss keeps decreasing. **DANN** exposes the same behaviour as the target loss of **AHDM** decreases.

σ_{shift}). This experiment helps to understand the purpose of unsupervised domain adaptation. Indeed, as the underlying condition for unsupervised DA to work is that the labelling function is similar in every domain ($\eta_{\mathcal{H}}$ small in our experiments), unsupervised DA is closely related to the issue of generalization. As a consequence, if the algorithm learnt on the source data is able to generalize, domain adaptation brings no improvement. As such, one can see unsupervised DA as a data-driven regularization to improve the target risk.

For this experiment, we use a shallow network with 2 layers with 5 neurons and *LeakyRelu* activation for the feature extractor and 1 layer for the final predictor. We keep this architecture for three methods: Multi-Layer Perceptron (**MLP**) without adaptation, Domain-Adversarial Neural Network (**DANN**) minimizing the $d_{\mathcal{H}}$ between domains and our Adversarial Hypothesis-Discrepancy Adaptation (**AHDM**) our method in the case of single source (no weights α). To get the results for **DANN**, we tune then hyperparameter μ balancing the regression and domain losses: without this tuning, **DANN** always fails to converge because its adversarial scheme is related to classification.

We conduct the experiments with various amount of shift. In Figure 4.6, we report the validation loss (computed validation set different from the training set) and target loss for each method for $\sigma_x = 0.2$, $\mu_{shift} = 0.5$ and $\sigma_{shift} = 0.5$, which is the case of target data related but not too close to the source domain.

While **MLP** and **DANN** overfit on the source data, our method is able to decrease the target loss. Hence, our adversarial scheme helps the algorithm to better learn for the target data. We report in Table 4.1 the average MSE scores over ten runs for the three methods and various shifts. One can notice that the further μ_{shift} is, the more useful the adaptation is.

We also report in Figure 4.7 a visualization of the extracted features from **AHDM** and **DANN** which shows that **DANN** tries to align domains only to be able not to separate them while **AHDM** is constrained by the final regression task.

μ_{shift}	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
MLP	**0.221**	**0.214**	0.260	0.360	0.510	0.695	0.909	1.132	1.322	1.484	1.632
DANN	0.223	0.235	0.296	0.412	0.581	0.784	1.017	1.258	1.462	1.629	1.772
AHDM	0.222	**0.214**	**0.255**	**0.350**	**0.490**	**0.670**	**0.881**	**1.044**	**1.197**	**1.392**	**1.446**

Table 4.1: Single source domain adaptation: MSE for different amounts of shift.

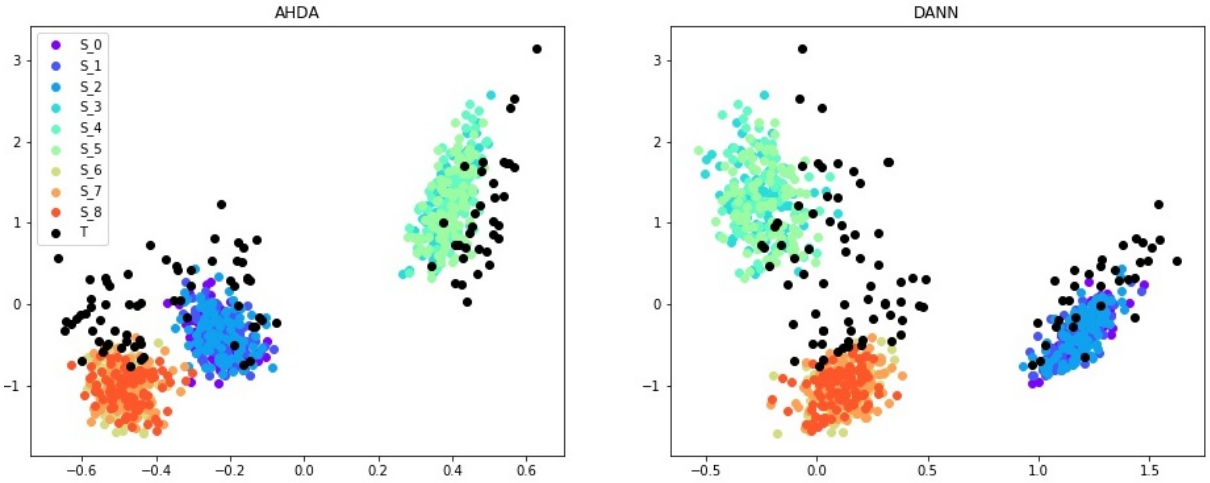
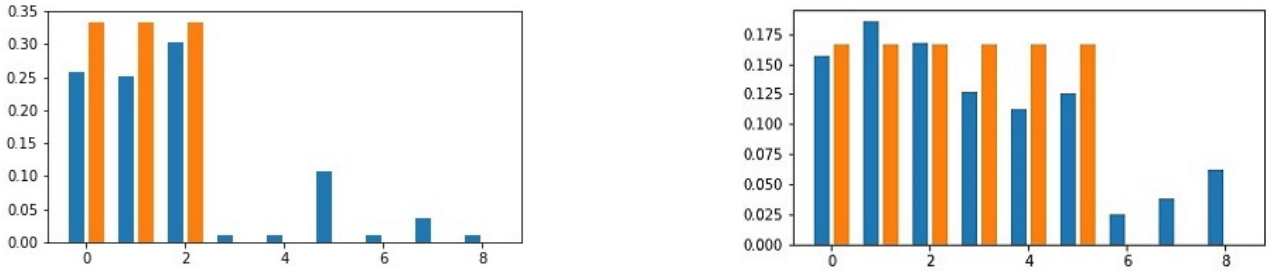


Figure 4.7: X-axis: Extracted features (before the final predictor) using **AHDM** (left) and **DANN** (right) ; Y-axis: labels to predict (y)

Multi-Source DA We also evaluate our multi-source algorithm on the previous target domain. As expected, multi-source domain adaptation does not bring any improvement in the previous scenario as the target is sharing the same relations with every source. But MSDA is particularly interesting in the case where some of the sources are not useful for adaptation. We demonstrate it in an experiment where we control the weights α given to each source in the creation of the target domain.

In a first experiment, we attribute equal weights $\alpha_{1,2,3} = 1/3$ to every source in one cluster and $\alpha_{4:9} = 0$ for every other cluster. In a second experiment, we attribute equal weights to two clusters $\alpha_{1:6} = 1/6$ and $\alpha_{7:9} = 0$ In both experiments, and on different runs, the algorithm is able to retrieve weights close to the real ones as illustrated in Figure 4.8. It is translated by a decrease of the target loss. When putting weights to only one source in a cluster, our algorithm struggles to identify a specific source but still gives large weight to sources in the same cluster. In some experiments where sources were very different from each other (no cluster), we notice a tendency for α to give 1 for one source and 0 for all the others. It can be meaningful as the target may be close to only one source in that case or be balanced using the λ parameter of the ℓ^2 regularization.



(a) One cluster with uniform weights

(b) Two clusters with uniform weights

Figure 4.8: Friedman Multiple Source experiment: α found by **AMSHDM** (blue) vs True α (orange)

Discussion Based on this toy experiment, we conclude that our **AMSHDM** and **AHDM** perform well for regression under the condition that the labelling functions are the same. The proposed weighting scheme is performing well when sources have really different relations to the output. The main limitation to $HDisc$ is that since it is dependent on h, its estimation is hard, especially in regression where values are not constrained. The weighting scheme mainly helps when the target data is close to only few of the sources. In the next experiment, we show how our algorithm performs on a real world dataset.

4.5.2 Appliance Consumption Estimation

Data Presentation

House	12	14	11	19	21	5	9	22
Number of days	138	233	496	245	119	238	512	85
Avg Daily Power (kWh)	16.45	14.02	8.55	7.12	6.10	4.16	3.90	3.72

Table 4.2: ElectricData: statistics of each house with a water heater

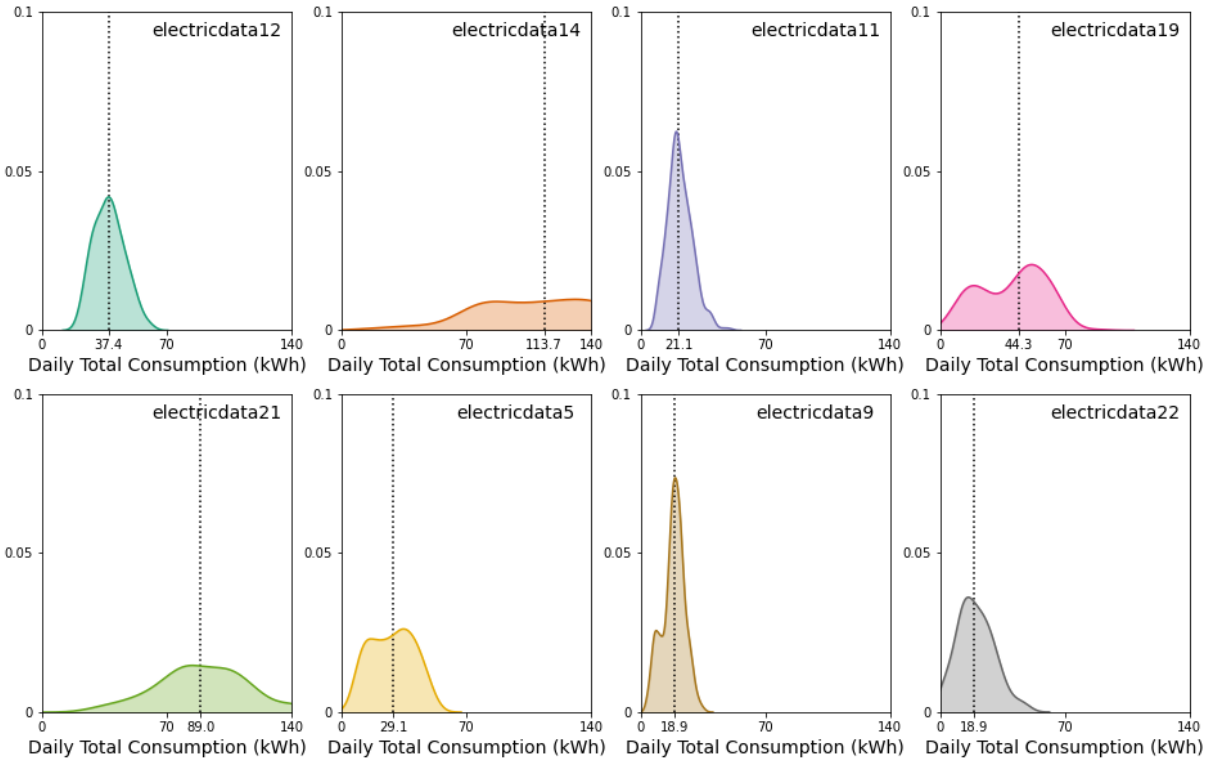
We apply our method on the problem of appliance consumption estimation: from the total consumption of a home during a day sampled every $10s$, can one predict the total consumption of a specific appliance. We focus our analysis on water heater, which is responsible to around a third of total consumption in many French households.

In the dataset, 8 houses with a water heater have been monitored (in total, more houses were monitored but we only keep the ones with clean data). We use data at a $10s$ sampling, which gives an input $X \in \mathbb{R}^{8640}$ and an output $y \in \mathbb{R}^+$. Each house has different characteristics as presented in Table 4.2.

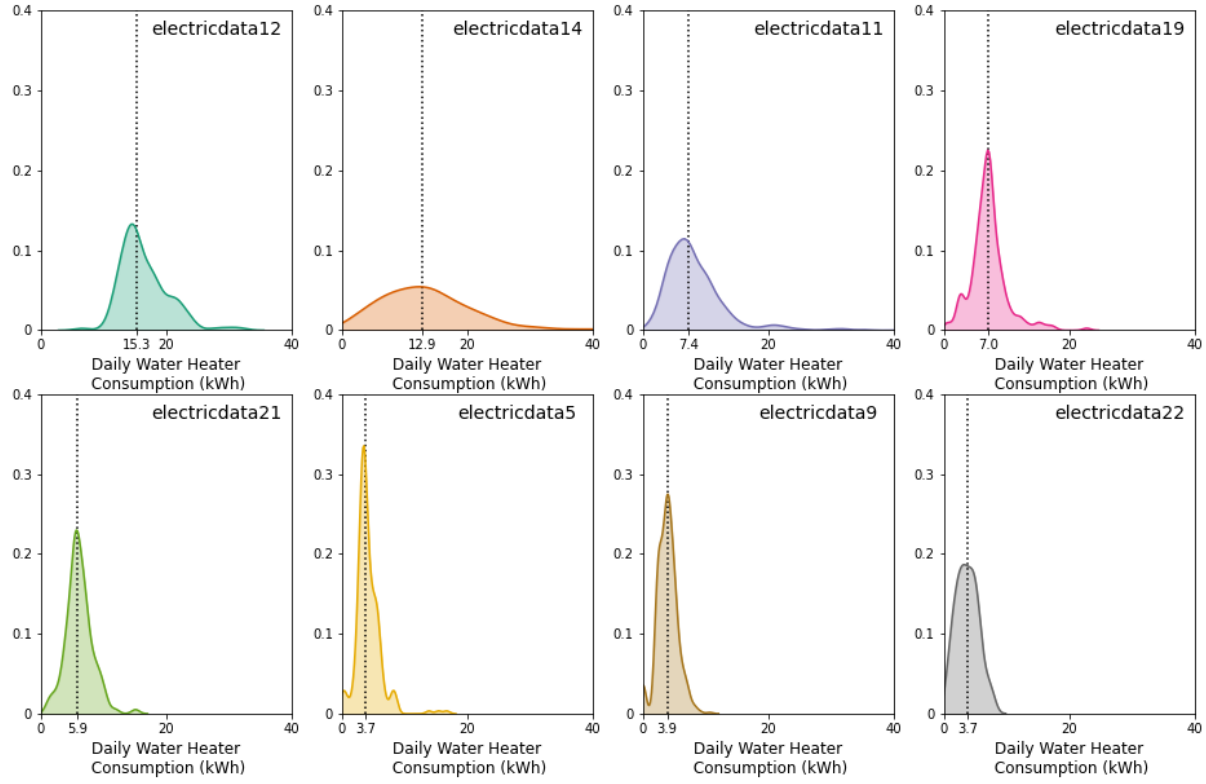
Moreover, the Transfer Learning problem is highlighted in Figure 4.9 where the distributions of daily total consumption and daily water heater consumption are represented. One can observe that the first two houses have a very high water heater consumption (around $15kWh$) whereas other houses have a smaller one.

Model

After trying different architectures of neural networks for our adaptation model, we noticed that the DenseNet architecture used in Chapter 3 underperforms compared to an architecture similar to Temporal Convolutional Networks [14] presented in Figure 4.10. Namely, our feature extractor is made of dilated causal convolutions defined in Section 3.2. Using dilated convolutions limits overfitting from the network and the causal convolutions allows for better performance. We use three causal convolution layers, with size 64, 128 and 256 and a dilation rate of 6 at each layer, and a dropout rate of 0.3 at each layer. BatchNormalization leads to poorer results, hence we do not include it in our architecture. We also include residual connections between each layers. Given the small size of the dataset, we expected overfitting from the network but found out that the residual connections mitigated this issue. Different normalizations have been tried for the input data as presented in the previous Chapter but did not find any improvement



(a) Total consumption distribution



(b) Water Heater consumption distribution

Figure 4.9: Distribution of Total and Water Heater consumption for each house (scales are the same inside sub-figures (a) and (b) respectively)

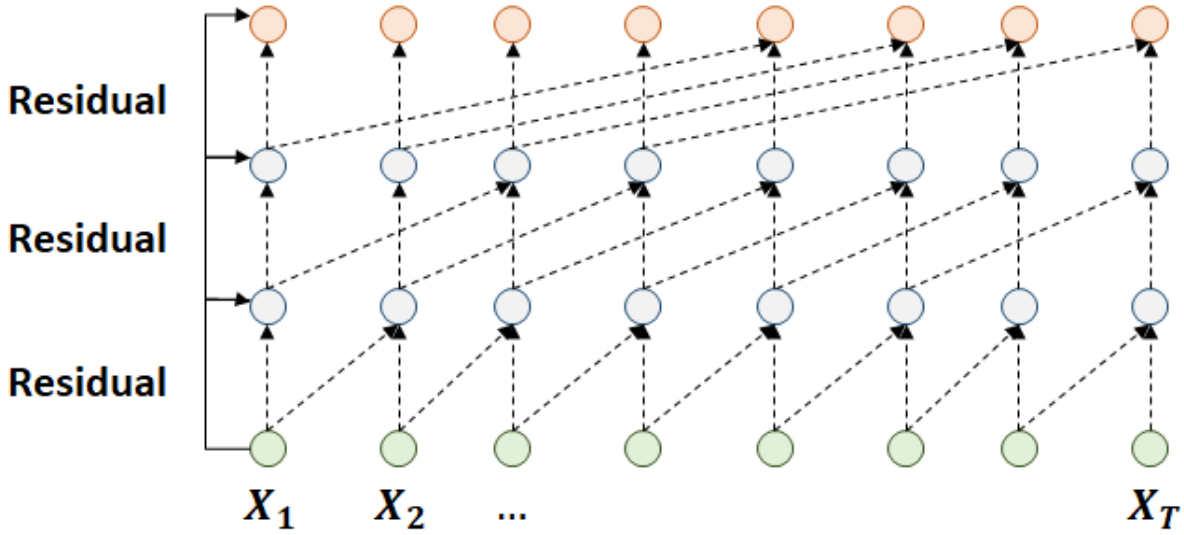


Figure 4.10: Temporal Convolutional Network model used in our experiments

with our ensembling, whereas it increased training time (details are given in Appendix C.4).

4.5.3 Same-house results

Firstly, our goal is to see the results in the same-house scenario, as a baseline result without adaptation. For each house, we firstly try to use K-fold cross-validation using only data coming from one house. We experiment three different methods:

- **LR:** linear regression from the total consumption to the water heater consumption.
- **RF:** extracting 113 statistical features (see Appendix C.2.1 feeded to a Random Forest (as in the appliance recognition experiment).
- **ResNet:** a convolutional neural network with residual connections, similar to the DenseNet presented in Chapter 3.
- **TCN:** the Temporal Convolutional Network presented above.

We keep the same architectures for every house. For the ResNet, we use 3 layers with kernel sizes 64, 128 and 256 and a dropout rate of 0.3. The main difference between the ResNet and the TCN is the causal convolution.

In Table 4.3, we give the average MAE over 5 runs for each house. We see that **TCN** gives the best results on almost all houses. We also notice that **TCN** tends to overfit. Indeed, for each house, only around 200 days are available for training, where each day is a time series of length 8640.

In an additional experiment, we use data from every house for training but only data from one house (that is seen during training) for test. We observe in Table 4.4 that both deep learning methods benefit from this additional data.

Method	LR	RF	ResNet	TCN	W.H. consumption
electricdata12	3.45	3.58	2.47	**2.25**	16.45
electricdata14	**4.29**	4.38	4.59	4.52	14.02
electricdata11	3.29	3.24	2.51	**2.38**	8.55
electricdata19	3.91	3.28	1.79	**1.68**	7.12
electricdata21	2.58	2.71	2.06	**1.76**	6.10
electricdata5	2.73	2.37	1.52	**1.46**	4.16
electricdata9	2.58	1.84	1.48	**1.24**	3.90
electricdata22	2.23	1.99	1.73	**1.49**	3.72

Table 4.3: Average MAE (kWh) over 5 runs for each method and house for the same house experiment. In the last column, we report the total water heater consumption.

Method	ResNet (1 house)	TCN (1 house)	ResNet (every house)	TCN (every house)
electricdata12	2.47	**2.25**	2.58	2.61
electricdata14	4.59	4.52	3.61	**3.49**
electricdata11	2.51	2.38	2.42	**2.34**
electricdata19	1.79	**1.68**	1.87	**1.68**
electricdata21	2.06	1.76	1.87	**1.72**
electricdata5	1.52	1.46	1.48	**1.42**
electricdata9	1.48	1.24	1.37	**1.19**
electricdata22	1.73	1.49	1.52	**1.47**

Table 4.4: Average MAE (kWh) over 5 runs for each method and house for the same house experiment, using every house for training

Overall, in our experiments, we show that **TCN** gives the best results. We also notice that adding data from other houses improves the results. But this scenario is only possible when the test house is monitored, whereas in general, we do not monitor the water heater consumption. In the next section, we experiment our adaptation method for this new scenario.

4.5.4 Cross-house results

In this experiment, we assume that at training time, labeled samples from 7 houses and an unlabeled sample from the test house are available. At test time, a additional unlabeled sample from the test house is available.

Using our analysis of the previous section, we choose the **TCN** architecture presented in Section 4.5.2 as a feature extractor for every adaptation method. We compare different methods:

1. **TCN:** the TCN trained on the labeled samples from the training houses
2. **DANN:** DANN [72] with a regressor head for the predictor
3. **CORAL:** Deep CORAL [169]
4. **AHDM:** our method for the single-source scenario
5. **MDAN:** Multi-Source Adversarial Domain adaptation [196], which extends DANN to multi-source
6. **AMSHDM:** our method with the multi-source scenario

For every adversarial method, the feature extractor is a TCN, and the predictor and discriminator is a network made of 2 fully connected layers. For the single source method $(1\text{-}4)$, we merge data coming from every training houses to create a single source.

Method	TCN (1 source)	DANN (1 source)	CORAL (1 source)	AHDM (1 source)	MDAN (7 sources)	AMSHDM (7 sources)
electricdata12	4.78	4.87	4.51	4.38	5.28	**4.11**
electricdata14	5.62	5.98	4.89	4.82	6.39	**4.76**
electricdata11	3.12	3.28	2.68	2.71	2.88	**2.31**
electricdata19	1.89	1.97	1.79	1.73	1.92	**1.67**
electricdata21	3.46	3.32	2.95	2.93	3.62	**2.77**
electricdata5	1.90	2.05	**1.79**	1.81	1.86	1.80
electricdata9	2.29	1.96	1.60	1.87	2.30	**1.42**
electricdata22	2.12	1.99	1.79	1.87	2.04	**1.74**

Table 4.5: Average MAE (kWh) over 5 runs for each method and house for the cross-house experiment

The average MAE for every method and house is given in Table 4.5. One can see the improvement obtained with hypothesis-discrepancy based adaptation, whereas $\mathcal{H}$-divergence based adaptation gives poor results. Moreover, our weighting scheme gives a slight improvement: we notice that the weights tend to group houses with similar consumption levels. It can be seen in Figure 4.11, where we plot the weights obtained by **AMSHDM**.

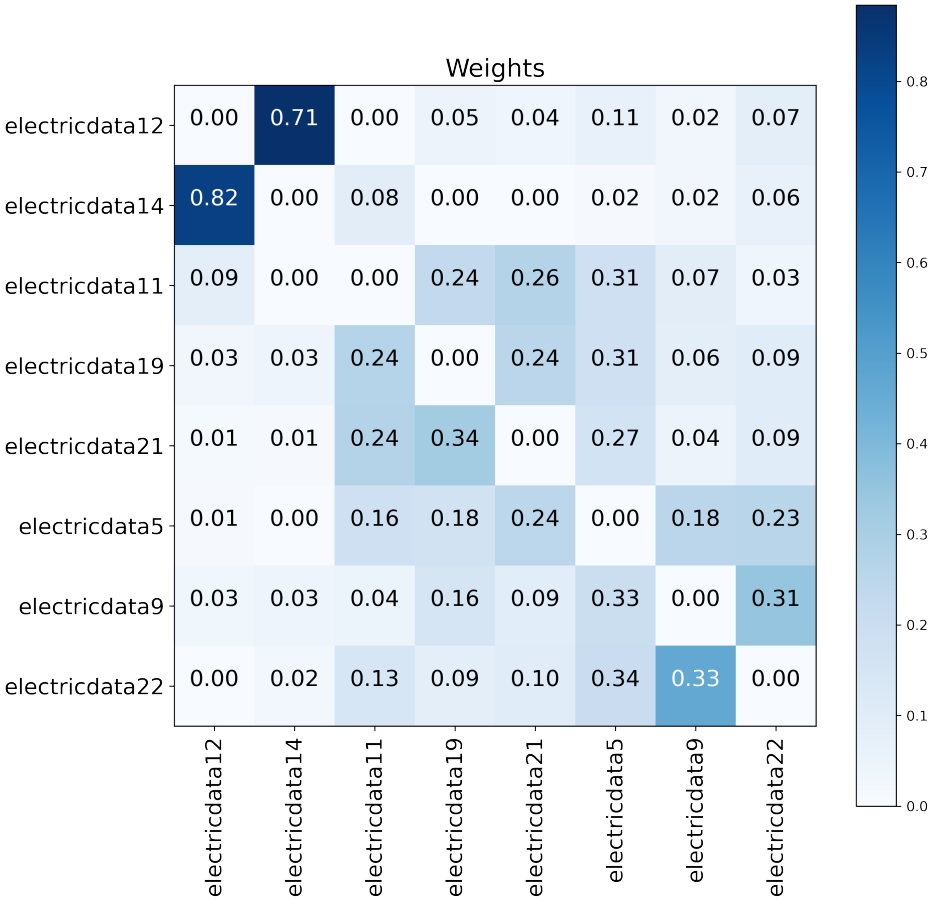


Figure 4.11: Weights found by **AMSHDM**

Finally, **CORAL** gives very good results. We expected it to be the case as **CORAL** regularization is similar to our hypothesis-discrepancy minimization as described in Section 4.3. As such, **CORAL** is a very good baseline as it is faster to train. Indeed, it only implies a regularization term instead of an adversarial scheme.

Comparison with data augmentation

The results highlights the efficiency of our method as it allows to get better results than not using adaptation. It must still be noted that similar results can be obtained using data augmentation. Indeed, a Multi-Agent Simulation (MAS) system has been developed at EDF to generate synthetic load curve of individual households: SMACH [7]. This tool is based on 10 minutes activity reports (27000 reports in total) which are representative of the french population's behaviour. Then for each activity, it helps to establish a sequence of actions for each profile in the MAS, defined by its duration, rhythm and preferential period of use. Based on the activities defined by those reports, the MAS outputs ON/OFF activations for each appliance in the household. Finally, for each activation, a signature is added at the timestamp.

Based on ElectricData, EDF researchers were able to generate synthetic data for a Transfer Learning problem. For each test house, they generated synthetic data using signatures from the other 7 houses (see Figure 4.13). More details about those experiments are available in [55]. The results obtained from training a **TCN** from synthetic data were slightly better than the ones obtained with **AMSHDM**.

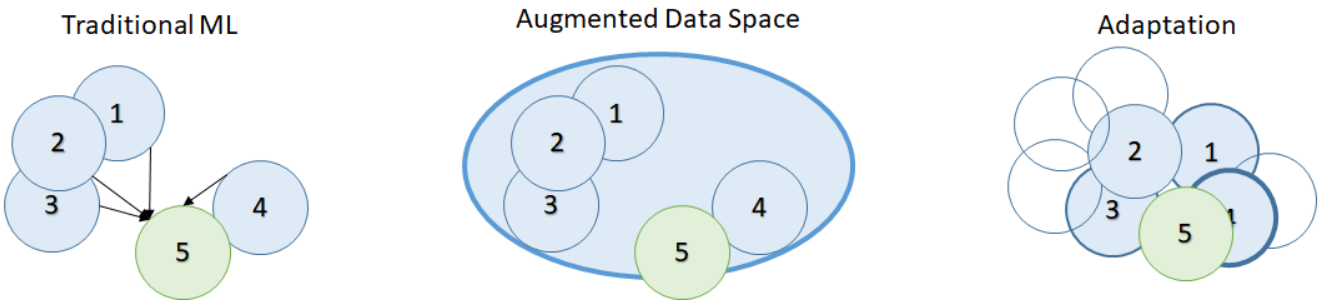


Figure 4.12: Data augmentation vs Adaptation: with data augmentation, a new training space is created using sources 1 to 4 ; with adaptation, the training data is moved closer to the testing data.

As such, data augmentation allows for better generalization. Our method allows to adapt to a specific space, as illustrated in Figure 4.12. The data augmentation still requires a lot of expert knowledge. Indeed, the MAS took years to develop and is not easy to use. We tried to use data augmentation based on GANs but got very poor results. This study opens the perspective of data augmentation driven by transfer learning. In the future, if large datasets are available, data augmentation seems to be a better option than Transfer Learning but data collection is difficult for NILM.

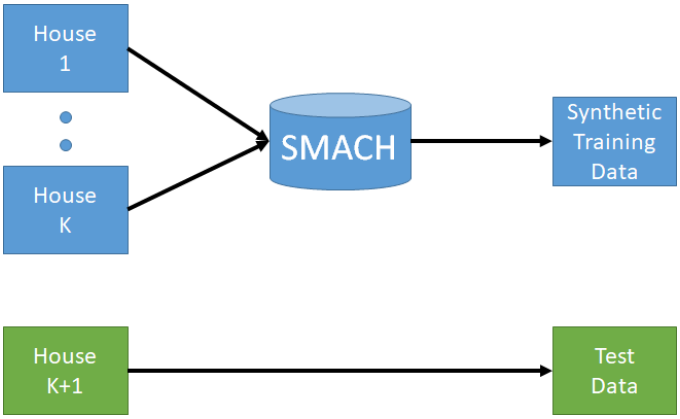


Figure 4.13: Learning from Synthetic Data

Other devices

We also try to transfer for other devices. It led to less interesting results overall as they were either too easy (fridge) or too hard (small appliances). For some appliances, we did not have enough data available to run our experiments. For instance, for the washing machine, we had on average only 18 days where it was turned on for the 16 houses where it was monitored. For the fridge, simply using a linear regressor from total daily consumption led to the best results. EDF experts agree that electric heater is very hard to study as it has a very high consumption and is very noisy. Because of those issues, testing for cross-dataset was impossible as electric water heater is barely present in other countries.

4.5.5 Experiments on other datasets

We also led some experiments on other datasets as our method is generic to any neural network architecture and can be applied to image or text for instance. We present experiments on two public domain adaptation datasets: (i) Multi-Domain Sentiment Dataset[2] (ii) Digit dataset made of different databases of images of digits.

Amazon Sentiment Dataset

We use the extended version of the Multi-Domain Sentiment Dataset[3] with 25 categories (books, dvd, ...). The dataset is made of textual reviews from Amazon and associated ratings. We treat it as a regression problem where the goal is to predict the rating based on the review. As in previous papers [196] [72], we transform it using tf-idf transform and filtering only the $5,000$ words with highest coefficients. Similarly to the original dataset, we keep $2,000$ samples for each category (or less when there was not $2,000$ available). We alternate on each category as a target and use every other categories as the multiple sources.

We compare our method to several other baselines:

[2] https://www.cs.jhu.edu/~mdredze/datasets/sentiment/
[3] https://www.cs.jhu.edu/~mdredze/datasets/sentiment/

- **MLP** corresponds to merging all sources and applying to the target dataset without adaptation.
- **DANN** corresponds to merging all sources into one and applying the **DANN** from [72].
- **AHDM** corresponds to merging all sources into one and applying our adversarial scheme using discrepancy specific to the task.
- **MDAN** is a multi-source domain adaptation based on the $d_{\mathcal{H}}$ distance proposed in [196]. It gives large weights to sources far from the target (we used the soft version).
- **AMSHDM** is our method described in Algorithm 1.

For **DANN** and **MDAN**, predictors have been specified to regression by changing the last head by a regression layer and loss with mean squared error. The implementation of **MDAN** is inspired from the original implementation from the authors. [4]. For fair comparison, the basic MLP architecture is kept the same for every method (see Appendix C.4).

Hyperparameter selection in unsupervised domain adaptation is a hard task as no labeled target data is available. For **DANN** and **MDAN**, we tried different sets of values and only report the ones giving the best results on test data which is very advantageous and normally not possible for unsupervised DA ($\lambda = 0.01$ for **DANN**, $\gamma = 10$, $\mu = 0.1$ for **MDAN** which are the hyperparameters they used in the classification setting). For our algorithm, the only hyperparameter is λ for ℓ^2-regularization of α for which we tried different settings and we report results for $\alpha = 0.01$ and $\alpha = 1$ here.

We run the experiment 5 times and report the average MAE for each method in Table 4.6. For most domains, either single-source or multi-source **AHDM** obtains the best result with multi-source bringing an improvement for many domains. One can notice the instability of **DANN** and **MDAN** for a few number of domains where the error becomes very large. Overall, the **AMSHDM** gets state of the art result and shows its power over $d_{\mathcal{H}}$ for adversarial regression domain adaptation. The weights obtained by our method were meaningful for some domains (giving large weight for software to transfer to video games for instance) but not for all.

Digits

Finally, we experiment our method on a digit classification task using 5 different domains: *MNIST*, *MNIST-M*, *Synth*, *SVHN* and *USPS*. *SVHN* and *USPS* are known to be the hardest datasets to classify as they are much more diverse. In each experiment, we use one domain as the target and the 4 others as the sources. We resize all domains to images made of 28×28 pixels. For fair comparison we use the same architecture for every network, using a simple convolutional neural network (see Appendix C.4 for the detailed architecture).

[4] https://github.com/KeiraZhao/MDAN

Dataset	apparel	auto	baby	beauty	books	camera	cellphones	computer	dvd
No-adapt	0.897	0.902	0.841	0.880	1.024	0.923	0.871	0.842	0.912
AHDM	**0.837**	0.887	0.840	0.860	0.945	0.893	**0.859**	**0.849**	0.882
DANN	0.929	1.017	0.893	0.878	1.141	0.944	1.051	1.135	1.05
MDAN	0.980	**0.797**	0.908	0.973	1.234	0.921	0.954	1.749	1.322,
AMSHDM	0.847	0.930	**0.814**	**0.746**	**0.928**	**0.853**	0.865	**0.849**	**0.786**
Dataset	**electronics**	**food**	**grocery**	**health**	**jewelry**	**kitchen**	**magazines**	**music**	**musical**
No-adapt	0.833	0.882	**0.774**	0.869	0.759	0.851	0.960	0.976	0.778
AHDM	0.844	0.866	0.796	0.851	0.815	**0.849**	0.909	0.955	0.895
DANN	1.064	1.036	0.793	0.955	0.783	0.856	0.985	1.293	**0.727**
MDAN	1.041	0.820	0.789	0.987	**0.755**	0.850	1.295	1.330	1.117
AMSHDM	**0.823**	**0.851**	0.838	**0.832**	0.767	0.853	**0.828**	**0.865**	0.829
Dataset	**office**	**outdoor**	**software**	**sports**	**tools**	**toys**	**video**	**Average**	**Avg rank**
No-adapt	0.969	0.831	0.924	0.844	0.823	0.873	0.866	0.892	3.00
AHDM	0.956	0.851	0.880	0.839	**0.813**	0.864	0.878	0.868	2.68
DANN	0.854	0.803	1.090	0.850	0.888	0.861	1.00	0.955	3.84
MDAN	1.041	**0.789**	0.964	0.910	1.656	0.843	1.467	1.06	3.88
AMSHDM	**0.923**	0.827	**0.846**	**0.823**	0.844	**0.838**	**0.751**	**0.838**	**1.60**

Table 4.6: Average MAE over 5 runs for each method and domain of the Amazon Multi-Domain Dataset

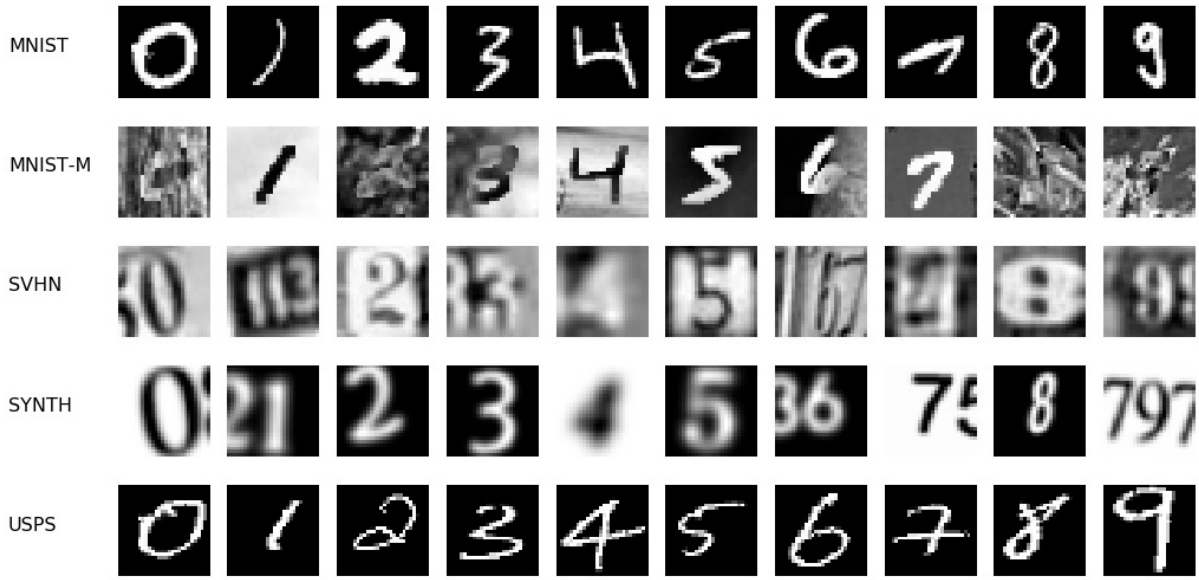


Figure 4.14: Visualization of digits datasets

For this experiment, we use a reduced version of the datasets with $10,000$ samples in each domain. Since we are in a classification setting, the loss used to train **AHDM** cannot be the Mean Squared Error. For the bound to hold, we need a loss that is symmetric and follows the triangle inequality. However, we found that the cross-entropy loss is performing better than the ℓ^1-loss, hence we use it to compute the discrepancy. Our adversarial structure becomes very close to the Maximum Classifier Discrepancy introduced in [158] and our weighting scheme extends this structure to a multi-source setting.

We report the accuracy for each dataset and each method in Table 4.7. One can observe that for a classification task, our $HDisc$ still provides good results even though there is no clear improvement compared to methods tailored for classification. For MNIST, even without adaptation, the CNN can learn general features from other datasets. **MDAN** fails as it gives large weights to datasets that are not similar (such as **USPS** for **SVHN**) while our **AMSHDM** does not suffer from this drawback.

Dataset	MNIST	MNIST-M	SVHN	Synth	USPS
CNN	0.916	0.450	0.489	0.562	0.624
DANN	0.918	**0.530**	**0.546**	0.710	0.659
AHDM	0.912	0.512	0.510	0.769	**0.667**
MDAN	**0.923**	0.460	0.488	0.671	0.574
AMSHDM	**0.923**	0.518	0.501	**0.793**	0.657

Table 4.7: Accuracy for the visual adaptations on digits datasets

4.6 Extension to semi-supervised adaptation

Method

In our work, we assume that the labelling functions are similar in every domain. But it is known not to be the case in many scenarios. When it is not the case, two solutions are available: (i) assume further knowledge about the structure of the labelling functions (ii) collect some labels on the target domain to perform semi-supervised domain adaptation.

We propose a simple solution in the second case to perform semi-supervised domain adaptation using hypothesis-discrepancy. In this framework, the only difference is the presence of a small labeled target sample $\mathcal{S}_T = \{(x_T^{(1)}, y_T^{(1)}),$ $..., (x_T^{(n_T)}, y_T^{(n_T)})\}$. Our solution is motivated by the $\mathcal{Y}$-discrepancy introduced in [128] which is an extension of the original discrepancy to the semi-supervised scenario defined by

Definition 11. *For two distributions p_S and p_T with associated labelling functions f_S and f_T, the $\mathcal{Y}$-discrepancy is defined as*

$$disc_{\mathcal{Y}}(p_S, p_T) = \sup_{h \in \mathcal{H}} |\mathbf{R_S}(h, f_S) - \mathbf{R_T}(h, f_T)| \tag{4.20}$$

In semi-supervised domain adaptation, $\mathcal{Y}$-discrepancy can be estimated with finite samples, even though its estimation is limited by the small size of the labeled target sample. It is easy to see that

$$\mathbf{R_T}(h, f_T) \leq \mathbf{R_S}(h, f_S) + disc_{\mathcal{Y}}(p_S, p_T) \tag{4.21}$$

As such, this bound is not very informative as it is a simple implication of the triangular inequality. We still tried to derive an adversarial scheme using $\mathcal{Y}$-discrepancy. We see in Equation 4.21 that in order to minimize $\mathbf{R_T}(h, f_T)$ we can formulate a min-max objective as before, by introducing a feature extractor ϕ_θ and a class of predictor $\mathcal{H}_\mathcal{Z}$:

$$\min_{\phi_\theta, h \in \mathcal{H}_\mathcal{Z}} \max_{h' \in \mathcal{H}_\mathcal{Z}} \mathbf{R_S}(h \circ \phi_\theta, y_S) + \mathbf{R_T}(h \circ \phi_\theta, y_T) + |\mathbf{R_T}(h' \circ \phi_\theta, y_S) - \mathbf{R_S}(h' \circ \phi_\theta, y_T)| \tag{4.22}$$

In this formulation, h tries to give low error rate on both source and target data. We add the term $\mathbf{R_T}(h \circ \phi_\theta, y_T)$ even though it is not present in the bound as it can only help h to be a better predictor for target data. On the other

hand, h' tries to maximize the $\mathcal{Y}$-discrepancy between source and target domains. Finally, the feature extractor minimizes a combination of source and target risks and the $\mathcal{Y}$-discrepancy.

Using this scheme allows the feature extractor to create features that are discriminative and where predictors cannot compute very different predictions between source and target data. This regularization prevents overfitting on source or target data, as shown in our experiments.

We use our weighting scheme as presented in the unsupervised scenario, by creating an α-weighted source distribution $p_\alpha = \sum_{k=1}^{L} \alpha_k p_k$ to the estimation of $Disc_{\mathcal{Y}}$:

$$Disc_{\mathcal{Y}}(p_\alpha, p_T) = \sup_{h' \in \mathcal{H}} \left| \sum_{k=1}^{K} \alpha_k \mathbf{R_k}(h', f_k) - \mathbf{R_T}(h', f_T) \right| \tag{4.23}$$

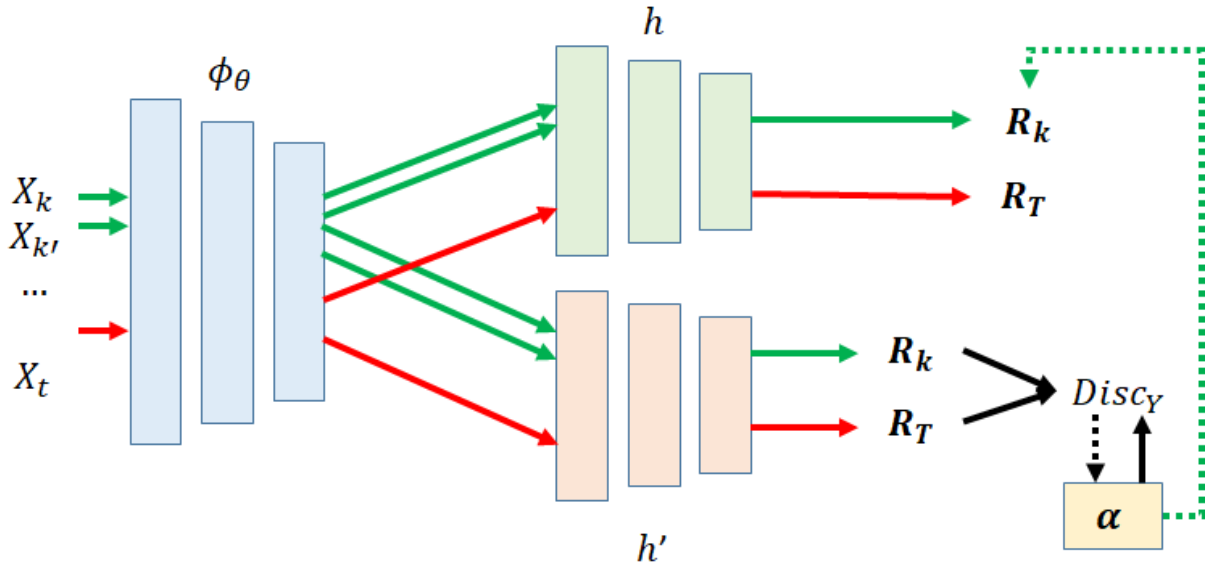


Figure 4.15: Adversarial Multi-Source $\mathcal{Y}$-Discrepancy Minimization (**AMSYDM**)

We represent in Figure 4.15 the global architecture of our semi-supervised adaptation **AMSYDM** (Adversarial Multi-Source $\mathcal{Y}$-Discrepancy Minimization). Then the different parameters of our models are updated alternatively:

1. $\mathcal{L}_h = \sum_{k=1}^{K} \alpha_k \mathbf{R_k} + \mathbf{R_T}$
2. $\mathcal{L}_{h'} = -Disc_{\mathcal{Y}}$
3. $\mathcal{L}_\theta = Disc_{\mathcal{Y}} + \sum_{k=1}^{L} \mathbf{R_k} + \mathbf{R_T}$
4. $\mathcal{L}_\alpha = Disc_{\mathcal{Y}} + \lambda \|\alpha\|_2$

We only consider the case where a small labeled target dataset was available without any unlabeled data, however, it is also be possible to extend the model for the case of unlabeled data.

Experiments

We run experiments on the Electric Data database for the water heater consumption estimation. We use the same experiment process as in Section 4.5.4, with the difference that we consider small labeled dataset for the target domain. Namely, we tried with target datasets of size 5, 10, 20 and 100.

In Table 4.8, we give the average MAE on every house obtained by different methods: **TCN-S** is directly learnt on every source without adaptation ; **TCN-S+T** is directly learnt from a combination from the source and target data without adaptation ; **AMSHDM** is performing unsupervised DA ; **AMSYDM** is the scheme described above. We keep the same basic architecture of the unsupervised DA.

Target size	5	10	20	100
TCN-S	3.15			
TCN-S+T	2.87	2.45	2.12	2.00
AMSHDM	3.08	2.97	2.75	2.57
AMSYDM	**2.78**	**2.21**	**1.98**	**1.84**

Table 4.8: Average MAE (kWh) over every house for the water heater consumption for different target sample size

As expected, the larger the target labeled sample size, the better is the performance on the target test sample. Moreover, we see that **AMSYDM** gives a small improvement on the no-adaptation framework. It also gives a notable improvement compared to unsupervised domain adaptation. It can be explained easily for this use case as the labelling functions of every domain are different. Yet, we assume it to be small in our domain adaptation guarantee (Theorem 11).

Limits

Several limitations prohibit the use of $\mathcal{Y}$-discrepancy in practice. The main one is the availability of a labeled target sample. For Non Intrusive Load Monitoring, it is costly to monitor a single appliance because the main cost comes from the installation of a sensor. Hence, in general, for the target domain, either no labeled data is available or a large number of labeled samples are available and there is no need for adaptation. The applicability of semi-supervised domain adaptation for non intrusive load monitoring is very restrained.

Moreover, $\mathcal{Y}$-discrepancy does not have better estimation guarantees than the target risk. Hence, there is no theoretical guarantee as to why our adaptation algorithm would give a better predictor for the target domain than a predictor trained using empirical risk minimization.

In [54] we propose a framework for re-weighting the source distribution using the $\mathcal{Y}$-discrepancy and neural networks. We formulate an objective that gives large weights to source samples that are close to the target samples in the sense of the $\mathcal{Y}$-discrepancy. This is still an ongoing work and opens perspectives for semi-supervised domain adaptation for regression.

4.7 Conclusion

In this chapter, we introduce a generic framework for adversarial domain adaptation with hypothesis-discrepancy. This framework encompasses some existing methods as we highlighted the links between hypothesis-discrepancy and existing works such as MCD or CORAL. After presenting the theoretical guarantees, we showed that unsupervised domain adaptation is useful for Non Intrusive Load Monitoring.

In general, our method give state-of-the art results on several regression domain adaptation problems and results slightly inferior to state-of-the art methods for classification, which seems to indicate that the approximation of $\mathcal{H}$-divergence proposed by Ben-David is better than our hypothesis-discrepancy in this case. For the specific problem of Non Intrusive Load Monitoring, we see that Transfer Learning is a hard task. Our method reduces the performance gap between same-domain learning and cross-domain transfer and provides good results on both ElectricData and Refit. We also proposed a framework for semi-supervised domain adaptation, that even though its guarantees are limited, led to good results in practice.

We identify two main possible ideas for future works. The first one is improving the interpretability of our method. When matching distributions learnt by a neural network, negative transfer can easily happen. Our weighting scheme give good results in practice but the convergence of the weights to ”good” weights is also not guaranteed as they are learnt through stochastic gradient descent. Moreover, neural networks require large amount of data to be trained, which is not available for many applications. This is often circumvented for image or text applications by using pre-trained neural networks which will hopefully be available for time series in the future.

The second idea comes from the comparison with data augmentation. The first experiments showed that using the SMACH multi-agent system with only 20 houses already led to a model that generalized better than one learnt on only real data. In the future, one could imagine collect more data and create the ImageNet of NILM using SMACH that would be useful for deep learning based methods.

Chapter 5

Covariance-based Transfer Learning with applications to Multivariate Time Series

Contents

The main applications of previous chapters focus on electricity consumption, one of the main part of EDF operations. However, the main mission of EDF is electricity production, with a large variety of production means such as nuclear or thermal power plants, power dams, wind farms, solar panels ... Those systems also bring a large number of machine learning challenges: for instance, wind farms and solar panels require production prediction as their production is not controlled. System monitoring is also required to plan maintenance in power plants.

For some systems, a large history of data is already available. For instance, most power plants have been implemented more than 20 years ago. Similarly, wind farms have been implemented during the last 15 years. When a new wind farm is implemented for production, it is natural to capitalize on existing wind farms to calibrate a model for failure prediction or consumption forecast. As technology evolves, there might be a shift between existing wind farms and new wind farms. Moreover, the environment is different for every farm which raises the issue of transfer learning.

This monitoring is often performed with sensors installed on some critical devices of the power plants, which leads to multivariate time series analysis. As opposed to uni-variate time series analysis, where temporal dependence is the most informative feature, multivariate time series analysis raises the issue of cross-sensor dependence.

The work of this final chapter deals with a transfer method for multivariate time series analysis. As the methods developed in the previous chapters can also be extended by using architectures specific to multivariate time series, we introduce a more interpretable method dedicated to multivariate time series. Intuitively, we assume that the main information contained in our sensors can be explained by autocovariance between sensors and we base our method on auto-covariance matrices of the multivariate time series. We assume that some relationships between sensors are invariant between domains and develop a method to infer those unknown invariant relationships.

Our work is based on covariance matrix analysis as covariance matrices lie on the manifold of Symmetric Positive Definite (SPD) matrices. After presenting this manifold, we propose a method to extract a subspace common to source and target domains based on covariance information: this subspace is defined as a linear combination of the original sensors and we experiment our method on Human Activity Recognition datasets.

5.1 Outline of the method

To analyze multivariate time series, a naive solution is to perform uni-variate time series computations on each dimension independently, and then merge the obtained results. It is sub-optimal as it completely ignores the potential

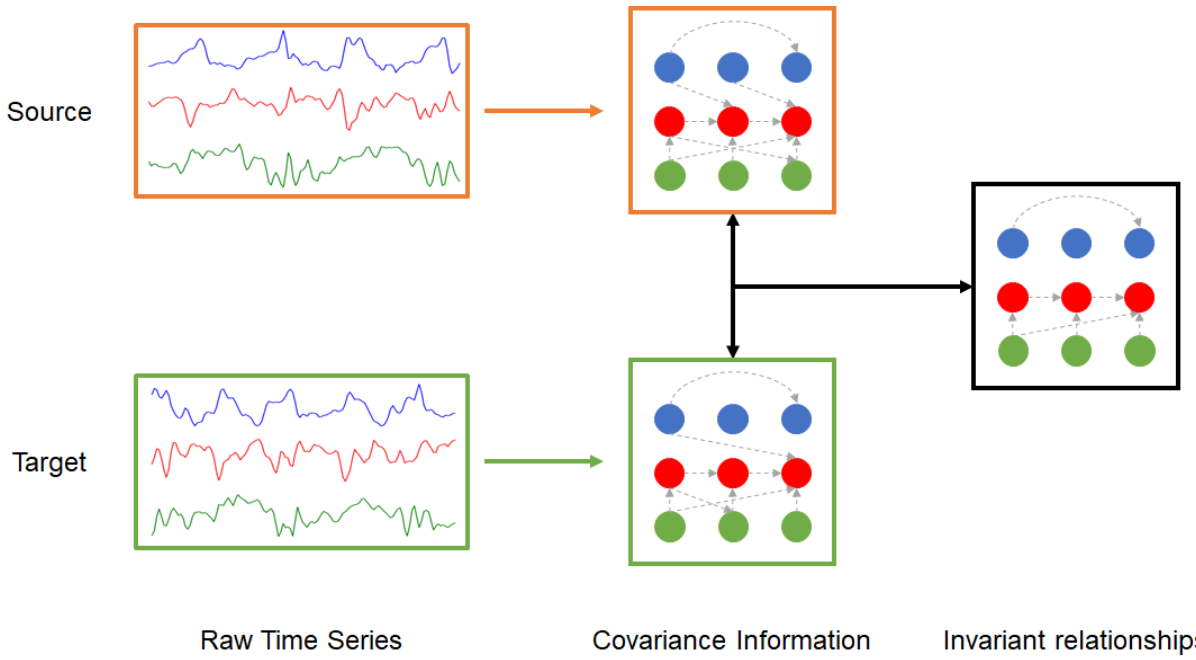


Figure 5.1: Illustration of the proposed method: raw time series are transformed to covariances and invariant relationships between sensors with different lags are extracted

dependence between dimensions, which is often very informative. Hence, we propose a method taking into account the relationships between dimensions.

Our idea is illustrated in Figure 5.1. We consider one or several source domains, for instance several existing wind plants, monitored by a set of sensors for which a labeled source sample is available and a target domain. Our method follows the idea of subspace-based domain adaptation [67]: we propose to extract a subspace on the set of sensors invariant between source and target domains. Our method is made of three steps:

1. Transform the raw time series into covariance or auto-covariance matrices, which are good features to represent relationships between sensors.

2. Learn a common subspace based on an objective composed of three parts aiming to:

   - Discriminate between labels (classes) ;
   - Align source and target domains in an unsupervised way (marginal distributions) ;
   - Align conditional source and target distributions.

3. Use the extracted subspace to learn a classifier on the source domain that can be generalized to the target domain.

As such, the goal of our method is to extract relationships between sensors that are invariant in every domain. We formulate the relationships between sensors using covariance matrices, which have a specific structure as they

are Symmetric Positive Definite (SPD). SPD Matrices lie on a Riemannian manifold [27] associated with metrics with interesting properties, finding applications in computer vision [83] or Brain Computer Interface [153]. We also use this geometry in our method and briefly present it in Section 5.3, focusing mainly on the advantage of these metrics when dealing with time series. The second step is to learn a common subspace of sensors using covariance matrices. For this purpose, we formulate a dimensionality reduction problem based on Maximum Mean Discrepancy (MMD) to align domains.

5.2 Multivariate Time Series and Covariance

The first step our method transforms the multivariate time series obtained with the sensors into covariance matrices. Here, we recall the definition of covariance and auto-covariance matrices for time series and their link with the popular Vector Auto-Regressive (VAR) model.

Covariance To deal with multivariate time series, auto-regressive models have been extended to vector auto-regressive models (VAR) where the coefficients represent the relationships between each dimension with lags. This is related to the notion of covariance and autocovariance. In the following, we consider a time series $X(t)$ of dimension t, sampled between 1 and T:

$$X = \begin{pmatrix} X_{1,1} & ... & X_{1,T} \\ ... & X_{i,t} & ... \\ X_{d,1} & ... & X_{d,T} \end{pmatrix}$$

indexed by $t \in \{1, ..., T\}$ with d the number of sensors. In the following, we also sometimes note as the value of X at time t by $X(t)$. Then the covariance matrix C is defined as

$$C = \frac{1}{T(T-1)} \begin{pmatrix} \mathbb{E}[X_1(t)X_1(t)] & ... & \mathbb{E}[X_1(t)X_d(t)] \\ ... & ... & ... \\ \mathbb{E}[X_d(t)X_1(t)] & ... & \mathbb{E}[X_d(t)X_d(t)] \end{pmatrix} \tag{5.1}$$

where $\mathbb{E}[X_i(t)X_j(t)] = \frac{1}{T-1}\sum_{i=1}^{T}(X_{i,t} - \bar{X_i})(X_{j,t} - \bar{X_j})$ ($\bar{X_i} = \frac{1}{T}\sum_{t=1}^{T} X_{i,t}$). C is the standard covariance matrix that does not take into account temporal dependencies. To better take into account the temporal dependence, we introduce the autocovariance matrix AC_w of x at a fixed horizon $w > 0$ as

$$AC_w = \begin{pmatrix} \mathbb{E}[X(t)^T X(t)] & \mathbb{E}[X(t-1)^T X(t)] & ... & \mathbb{E}[X(t-w)^T X(t)] \\ \mathbb{E}[X(t)^T X(t+1)] & \mathbb{E}[X(t+1)^T X(t+1)] & ... & \\ ... & ... & ... & ... \\ \mathbb{E}[X(t+w)^T X(t)] & ... & ... & \mathbb{E}[X(t+w)^T X(t+w)] \end{pmatrix} \tag{5.2}$$

where $\mathbb{E}[X(t+i)^T X(t+j)]$ is the covariance matrix between sensors with lags (assuming the time series are centered). Under the assumption that the time series is stationary AC_w is symmetric definite positive. In general, we assume wide-sense stationarity which means that the mean and autocovariance of the time series stays constant.

In most industrial systems, the stationarity assumption does not hold over the entire time series. However, a common way to circumvent it is to consider small time windows around an event of interest. On this small time window, stationarity is often ensured [116].

Partial Autocorrelation Covariance matrix and partial autocorrelation matrix are directly linked. Partial autocorrelation at order r of a time series X is defined as follows:

$$\rho(r) = \frac{\mathbb{E}[X(t)^T X(t-r)|X(t-1), ..., X(t-r+1)]}{\mathbb{E}[X(t)^T X(t)|X(t-1), ..., X(t-r+1)]\mathbb{E}[X(t-r)^T X(t-r)|X(t-1), ..., X(t-r+1)]} \tag{5.3}$$

Those coefficients estimate the relationship between values of X at time $t-r$ and time t after removing the effects of every intermediate value. They are also directly linked to the VAR coefficients. It can be shown that, denoting $\rho = \begin{pmatrix} \rho(0) & \rho(1) & ... & \rho(w) \\ \rho(1) & \rho(0) & ... & ... \\ ... & ... & ... & ... \\ \rho(w) & ... & ... & \rho(0) \end{pmatrix}$ and $P = AC_w^{-1}$ the precision matrix, we have

$$\rho_{i,j} = -\frac{p_{ij}}{\sqrt{p_{ii}p_{jj}}} \tag{5.4}$$

As highlighted in [81], partial auto-correlation matrix shows the conditional independence structure between the variables. It can then be used to detect causality between sensors with the Granger causality test. Moreover, Barndorff-Nielsen and Schou (1973) [19] proved that there is a one-to-one correspondence between partial correlations at horizon w and the coefficients of a Vector AutoRegressive model of horizon w. In [81], authors use the Partial Auto-Correlation Matrix as a feature to perform time series clustering.

Estimation When trying to estimate matrix AC_w (of horizon w) for a d-dimensional time series, the number of covariance coefficients to estimate is dw. When the time series length T is small, the estimation of AC_w is impossible. In the following, we use Ledoit-Wolfe regularization [108], which is better conditioned for high-dimensional

covariance estimation. But there still is a limit on the number of coefficients that we can estimate.

VAR Model The main method working on relationships between sensors and lags is the Vector AutoRegressive (VAR) model. A time series X is assumed to follow the VAR model of degree p if

$$X_t = \sum_{r=1}^{p} A_r X_{t-r} + e_t \tag{5.5}$$

where e_t is an error term such that (i) $\mathbb{E}[e(t)] = 0$ (ii) $\mathbb{E}(e_t^T e_t) = \Sigma$ (iii) $\mathbb{E}[e(t)e(t-s)] = 0$ for all $s > 0$. The first two conditions give the structure of the same-time dependence. The last condition assumes no correlation between different error terms.

The coefficients $A_r \in \mathbb{R}^{d \times d}$ can be directly estimated using Yule-Walker equations [181]. The VAR coefficients are the roots of a polynom whose coefficients are the autocovariance. Hence, the autocovariance matrix AC_w contains every information required for VAR model estimation.

Overall, autocovariance information is useful for modelling linear relationships between sensors at different timestamps. It has shown its efficiency in forecasting, but also in anomaly detection or time series classification. Hence, we propose to use covariance information as features for multivariate time series analysis.

5.3 Riemannian Geometry of Symmetric Positive Definite Matrices and Time Series

As our method is based on covariance matrices, we can use the specific geometry developed for SPD matrices. We present the metrics of interest for our work and why their properties are relevant for time series. We do not present details on the Riemannian manifold of SPD matrices but interested reader can refer to the extensive work of Bhatia (2009) [27].

5.3.1 Basics

We note the set of Symmetric Positive Definite (SPD) Matrices $\mathbb{S}_+^{(d)}$ such that

$$\mathbb{S}_+^{(d)} = \{M \in \mathbb{R}^{d \times d} \; ; \; M^T = M \; , \; x^T M x > 0, \; \forall x \in \mathbb{R}^d, x \neq 0\} \tag{5.6}$$

Bhatia (2009) [27] proved that when endowed with an appropriate Riemannian metric, $\mathbb{S}_+^{(d)}$ is a smooth Riemannian manifold with non positive curvature [130]. This means that every neighbourhood of a matrix $M \in \mathbb{S}_+^{(d)}$ can be bijectively mapped to Euclidean space. While we do not develop the theory of Riemannian geometry, we can use

some tools for machine learning applications. In the following, we detail the specific distances of these manifolds as they are more informative for some applications.

A SPD Matrix M admits the following decomposition $M = Q^T \Lambda Q$ where $Q^T Q = 1$ and $\Lambda = \begin{pmatrix} \lambda_1 & 0 & ... & 0 \\ 0 & \lambda_2 & ... & 0 \\ 0 & 0 & ... & \lambda_d \end{pmatrix}$ with $\lambda_1 \geq ... \geq \lambda_d > 0$. We respectively define the matrix exponential and logarithm over $\mathbb{S}_+^{(d)}$ as

$$\begin{aligned} expm(M) &= Q^T \begin{pmatrix} exp(\lambda_1) & 0 & ... & 0 \\ 0 & exp(\lambda_2) & ... & 0 \\ 0 & 0 & ... & exp(\lambda_d) \end{pmatrix} Q \\ logm(M) &= Q^T \begin{pmatrix} log(\lambda_1) & 0 & ... & 0 \\ 0 & log(\lambda_2) & ... & 0 \\ 0 & 0 & ... & log(\lambda_d) \end{pmatrix} Q \end{aligned} \tag{5.7}$$

The two main metrics proposed for $\mathbb{S}_+^{(d)}$ are the affine invariant metric (AIM) [143] and log-euclidean metric (LEM) [11], both having different properties. The Affine Invariant Metric between two matrices $M_1, M_2 \in \mathbb{S}_+^{(d)}$ is defined as

$$\delta_{AIM}(M_1, M_2) = \|logm(M_1^{-1/2} M_2 M_1^{-1/2})\|_F \tag{5.8}$$

where $\|.\|_F$ is the Frobenius norm. AIM has two interesting invariance properties:

- $\delta_{AIM}(M_1^{-1}, M_2^{-1}) = \delta_{AIM}(M_1, M_2) \rightarrow$ Invariance to inversion
- For any invertible matrix $W \in \mathbb{R}^{d \times d}$, $\delta_{AIM}(W^T M_1 W, W^T M_2 W) = \delta_{AIM}(M_1, M_2) \rightarrow$ Affine invariance

Those properties make it an attractive tool to process SPD Matrices with machine learning. The main drawback is that δ_{AIM} is computationally expensive.

The Log-Euclidean Metric between two matrices $M_1, M_2 \in \mathbb{S}_+^{(d)}$ is defined as

$$\delta_{LEM}(M_1, M_2) = \|logm(M_1) - logm(M_2)\|_F \tag{5.9}$$

LEM also has invariance to inversion, but does not have full affine invariance. It is still invariant to rotation. If we define the set of rotation matrix $\mathcal{O}(d) = \{W \in \mathbb{R}^{d \times d}, W^T W = I_d\}$ where I_d is the identity matrix of size d, then the two following properties hold.

- $\delta_{LEM}(M_1^{-1}, M_2^{-1}) = \delta_{LEM}(M_1, M_2) \rightarrow$ Invariance to inversion
- $\delta_{LEM}(sW^T M_1 W, sW^T M_2 W) = \delta_{LEM}(M_1, M_2)$ for any rotation matrix $W \in \mathcal{O}(d)$ and $s \in \mathbb{R}^+$, $\rightarrow$ Invariance to rotation and scaling

LEM is only invariant to rotation and scaling but is easier to compute and manipulate.

Finally, we mention a last divergence that does not define a manifold but still has good properties for SPD Matrices: Stein-Divergence [167]. The Stein Divergence between two matrices $M_1, M_2 \in \mathbb{S}_+^{(d)}$ is defined as

$$\delta_S(M_1, M_2) = \log det\left(\frac{M_1 + M_2}{2}\right) - \frac{1}{2}\log det(M_1 M_2) \tag{5.10}$$

Stein-divergence shares the same invariance properties as AIM but is less expensive to compute.

The three metrics defined above share similar invariance properties. Moreover, using these metrics, it is possible to define means, geodesics and even Gaussian distributions [157] over $\mathbb{S}_+^{(d)}$.

5.3.2 Working with time series

Let us consider a multivariate time series $X \in \mathbb{R}^{d\times T}$, its autocovariance matrix with horizon w AC_w and its covariance matrix $C = \frac{1}{T-1}XX^T$. The geometry detailed above gives good properties to the analysis of time series: (i) equivalence between working on sensors and underlying source signals when they are linearly mixed (ii)

The invariance to rotation means that it is equivalent to work on the covariance matrix C or W^TCW (and equivalently for autocovariance matrix). This implies that it is equivalent to work on the original time series X or W^TX as $W^TCW = (W^TX)^TWX$. This property is relevant to applications where sensors capture phenomenons coming from different sources, as in Source Separation. If we assume that the observed sensors X capture information from d sources S via $X = WS$, then for two time series X_1, X_2, $\delta_{LEM}(X_1, X_2) = \delta_{LEM}(S_1, S_2)$.

The invariance to inversion means that working with C or C^{-1} is equivalent. We saw that the precision matrix C^{-1} is equivalent to the partial correlation between sensors, up to a normalizing factor: if we note ρ_{ij} the partial autocorrelation between sensors i and j, we have

$$\rho_{ij} = \frac{(C^{-1})_{ij}}{\sqrt{C_{ii}^{-1}C_{jj}^{-1}}} \tag{5.11}$$

This property can be extended to auto-covariance matrix by noting that AC_w is the covariance matrix of $\begin{pmatrix} X_{1:T-w} \\ X_{2:T-w+1} \\ ... \\ X_{w:T} \end{pmatrix}$ where $X_{j:T-w+j} = \{x_j, ..., x_T - w + j\}$.

This means that when using those metrics, we use features that are very informative on the true relationship between sensors. As there is an equivalence between partial auto-correlation and VAR coefficients, our method allows to work implicitly on VAR coefficients, which are used in many applications.

Finally, one could note that when using auto-covariance matrix, AC_w is not only SPD but also Block-Toeplitz.

SPD Block-Toeplitz matrices also lie on a manifold with a specific metric, as proved in [18]. This metric is very hard to compute and has led to limited applications so far, but it would be interesting to study its use in future work.

5.3.3 Statistical Learning with SPD Matrices

We give ourself a classification problem with a dataset $\{(C_1, y_1), ..., (C_n, y_n)\}$ where $C_i \in \mathbb{S}_+^{(d)}$ and $y_i \in \{0, 1\}$. Different solutions have been proposed to integrate the specific geometry of $\mathbb{S}_+^{(d)}$ in classical machine learning approaches.

The first line of geometry-aware algorithms directly used the distances defined above in a k-nearest neighbour algorithm [174]. The minimum distance to mean classifier follows a similar idea: first, class barycenters are learnt on training matrices and at test time, a test sample is assigned to the class whose barycenter is the closest [16].

The line of interest for our method considers kernel based methods. For instance, Barachant et al. (2013) [17] propose a linear kernel for AIM using the tangent space and Jayasumana et al. (2015) [98] study kernel-based algorithms for different metrics. They show that the log-euclidean metric can be used to define a gaussian kernel between two matrices $M_1, M_2 \in \mathbb{S}_+^{(d)}$:

$$k_{LE}(M_1, M_2) = e^{-\gamma \delta_{LE}(M_1, M_2)} \tag{5.12}$$

where γ is a parameter of the kernel. The Log-Euclidean Gaussian Kernel is positive definite for any $\gamma > 0$. Unfortunately, it is not the case when using the Affine Invariant Metric. For the Stein divergence, a positive definite gaussian kernel can also be defined for specific values of γ [167]:

$$k_S(M_1, M_2) = e^{-\gamma \delta_S(M_1, M_2)} \tag{5.13}$$

with $\gamma \in \{\frac{1}{2}, \frac{2}{2}, ..., \frac{d-1}{2}\} \cup [\frac{d-1}{2}, +\infty[$. Using one of those kernels, authors proposed different methods for classification, kernel SVM being the most popular. Kernels have also been used for sparse coding to decompose SPD Matrices, with applications to computer vision [84].

The aforementioned methods have been very popular in Brain Computer Interface applications, but have led to few applications on industrial systems. We argue that the invariance of those metrics are relevant for industrial monitoring.

5.4 Transferable subspace using Covariance information

5.4.1 Framework

In our framework, we consider K domains $\mathcal{D}_k = \{\mathcal{X}, P_k(X_k)\}$ with K associated tasks $\mathcal{T}_k = \{\mathcal{Y}, P_k(Y_k|X_k)\}$. The input data are multivariate time series $\mathcal{X} \subset \mathbb{R}^{d \times T}$. We also assume that we have K labeled samples $\mathcal{S}_k =$

$\{(X_k^{(1)}, y_k^{(1)}), ..., (X_k^{(n)}, y_k^{(n)})\}$ where $X_k^{(i)} \in \mathbb{R}^{d \times T}$ and $y_k^{(i)} \in \{-1, +1\}$. We also assume that we have a target domain $\mathcal{D}_T = \{\mathcal{X}_T, P_T(X)\}$ and a target task $\mathcal{T}_T = \{\mathcal{Y}_T, P_T(Y|X)\}$ with an unlabeled sample $\mathcal{S}_T = \{X_T^{(1)}, ..., X_T^{(m)}\}$.

For instance, the K sources could be K existing wind turbines and the target a wind turbine that just got launched. We aim to present a very general framework even though some special cases are studied. If $K = 1$, it corresponds to the single-source scenario. Our solution encompasses domain generalization and still works if the target sample is unavailable at training time.

5.4.2 Learning a subspace aligning domains

In general, when monitoring a system, many variables are monitored but only a few of them are useful. This led to variable selection or dimension reduction. Here, we propose a method to reduce the dimension of the original time series using only covariance information. The method is done in two successive steps: (i) learn a common subspace between one or several source domains and target domain (ii) use the extracted subspace to learn a classifier on the source domain that can be generalized to the target domain.

Our model is formulated as follows: we want to learn a mapping parametrized by a matrix $W \in \mathbb{R}^{d' \times d}$ where $d' < d$ that maps every time series $X \in \mathbb{R}^{d \times T}$ to $W^T X$. The criteria for this dimensionality reduction is that the new features need to be discriminative and transferable between domains. We use only covariance information on the new data space. For every domain $\mathcal{D}_k$ and sample $\mathcal{S}_k$, we write $C_k^{(i)} = X_k^{(i)}(X_k^{(i)})^T$. One can note that if $W \in \mathbb{R}^{d' \times d}$ is full rank, then $W^T C_k^{(i)} W$ is SPD, *ie* $W^T C_k^{(i)} W \in \mathbb{S}_+^{(d')}$. Hence, we constrain W to full rank matrices.

In the following, we learn the kernel parametrized by W as:

$$k_{LE}[W](C_1, C_2) = e^{-\gamma \delta_{LE}(W^T C_1 W, W^T C_2 W)} \tag{5.14}$$

We further define

$$\mathcal{K}[W] = \begin{pmatrix} \mathcal{K}_{11}[W] & \mathcal{K}_{12}[W] & ... & \mathcal{K}_{1R}[W] & \mathcal{K}_{1T}[W] \\ \mathcal{K}_{21}[W] & \mathcal{K}_{22}[W] & ... & \mathcal{K}_{2R}[W] & \mathcal{K}_{2T}[W] \\ ... & ... & ... & ... & \\ \mathcal{K}_{R1}[W] & \mathcal{K}_{R2}[W] & ... & \mathcal{K}_{RR}[W] & \mathcal{K}_{RT}[W] \\ \mathcal{K}_{T1}[W] & \mathcal{K}_{T2}[W] & ... & \mathcal{K}_{TR}[W] & \mathcal{K}_{TT}[W] \end{pmatrix} \in \mathbb{R}^{Kn+m \times Kn+m}$$

the kernel matrix over all samples where $\mathcal{K}_{rs}[W]$ is the kernel matrix between samples from domains r and s.

Our objective is made of three parts: (i) the common subspace must be able to discriminate between labels on the source domains (ii) it must align marginal distributions between source and target domains (iii) it must align conditional distributions between source and target domains. In presence of several source domains, the adaptation is done between the different source domains as well.

Discriminative Learning We use the idea of Kernel Alignment [43] and formulate the following objective

$$J_y(W) = -Tr(\mathcal{K}[W]Y^TY) \tag{5.15}$$

where $Y = [y_1^{(1)}, ..., y_1^{(n)}, y_2^{(1)}, ..., y_2^{(n)}, ..., y_K^{(1)}, ..., y_K^{(n)}, 0_m]$ is the full label matrix with m concatenated 0 for the unlabeled target sample. Note that in the case of multi-class classification, one can simply replace Y^TY by a matrix H defined by $H_{ij} = \begin{cases} +1 if y_i = y_j \\ -1 if y_i \neq y_j \end{cases}$ Minimizing this objective forces elements of $\mathcal{K}[W]$ to be close to 1 when the labels of two samples are the same and 0 when the labels are different. This objective corresponds to the classical supervised setting of kernel alignment. Here, we considered domains of the same size, but if some domains are too heavily represented, it might be necessary to normalize their contribution to the objective.

Aligning source and target domains We saw in Section 2.2.3 that Maximum Mean Discrepancy can be used to measure the difference between distributions embedded in a RKHS. We use the classical empirical estimate to measure the MMD between two samples from domains k and T:

$$MMD(\mathcal{S}_k, \mathcal{S}_T) = Tr(\mathcal{K}_{kT}[W]M) \tag{5.16}$$

where for $1 \leq i, j \leq n + m$ $M_{ij} = \frac{1}{n^2}$ if $1 \leq i, j \leq n$, $M_{ij} = \frac{1}{m^2}$ when $n + 1 \leq i, j \leq n + m$ and $M_{ij} = -\frac{1}{nm}$ otherwise. We want to find features that minimize the discrepancy between source and target domains. Hence, we formulate the following objective with pairwise-MMD:

$$J_a(W) = \sum_{r=1}^{K} Tr(\mathcal{K}_{kT}[W]M) \tag{5.17}$$

Note that in the case when $K = 1$, this objective is similar to most MMD-based methods. In our case, we simply extend it to multiple sources using pairwise-MMD. As we give equal weight to every source domain, we assume them to have equal contribution to the adaptation.

Conditional Source Alignment Finally, we can align conditional distributions on source domains as labeled samples are available. For two source domains $\mathcal{D}_k$ and $\mathcal{D}_s$ with $1 \leq r, s \leq R$, we can measure the Maximum Mean Discrepancy between conditional distributions $P(X|Y)$. As above, for each class $c \in \{-1, +1\}$, we define a regularization matrix $M_c^{(rs)}$ between domains. We denote $\mathcal{S}_{r,c} = \{x_k^{(i)} \in \mathcal{S}_k; y_k^{(i)} = c\}$ and $\mathcal{S}_{s,c} = \{x_s^{(i)} \in \mathcal{S}_s; y_s^{(i)} = c\}$.

$$M_c^{(rs)} = \begin{cases} (m_c)_{ij}^{(rs)} = \frac{1}{n_{r,c}n_{r,c}} & \text{if } x_i, x_j \in \mathcal{S}_{r,c} \\ (m_c)_{ij}^{(rs)} = \frac{1}{n_{s,c}n_{s,c}} & \text{if } x_i, x_j \in \mathcal{S}_{s,c} \\ (m_c)_{ij}^{(rs)} = -\frac{1}{n_{r,c}n_{s,c}} & \text{if } x_i \in \mathcal{S}_{r,c}, x_j \in \mathcal{S}_{s,c} \\ & \text{or } x_i \in \mathcal{S}_{s,c}, x_j \in \mathcal{S}_{r,c} \\ (m_c)_{ij}^{(rs)} = 0 & \text{otherwise} \end{cases} \tag{5.18}$$

Finally, we note $M^{(rs)} = \sum_{c\in\{-1,+1\}} M_c^{(rs)}$. We introduce the objective on conditional distributions as

$$J_c(W) = \sum_{r,s=1}^{R} Tr[\mathcal{K}_{rs}[W]M^{(rs)}] \tag{5.19}$$

We can simplify the notations by introducing a single matrix L for every MMD-regularizer. We note

$$L = \begin{pmatrix} L^{(11)} & L^{(12)} & ... & L^{(1R)} & L^{(1T)} \\ ... & ... & ... & .... & \\ L^{(R1)} & L^{(R2)} & ... & L^{(RR)} & L^{(RT)} \\ L^{(T1)} & L^{(T2)} & ... & L^{(RT)} & L^{(TT)} \end{pmatrix} \in \mathbb{R}^{Rn+m\times Rn+m}$$

For $r, s \in \{1, ..., R, T\}$, for $r \neq s$, if we note $M^{(rs)} = \begin{pmatrix} A^{(rs)} & B^{(rs)} \\ (B^{(rs)})^T & D^{(rs)} \end{pmatrix}$ then we define $L^{(rs)} = B^{(rs)}$ and $L^{(rr)} = \sum_{s=1}^{r} A^{(rs)}$.

Using the matrix L defined above, we can simplify the definitions of our objectives as

$$J_a(W) + J_c(W) = Tr(\mathcal{K}[W]L) \tag{5.20}$$

where $\mathcal{K}$ is the kernel matrix over every domain.

Objective Finally, we formulate our objective as

$$\begin{gathered} \min_{W\in\mathbb{R}^{d\times d'}} J(W) = Tr(\mathcal{K}[W](\lambda L - Y^T Y)) \\ \text{such that } W^T W = I_{d'} \end{gathered} \tag{5.21}$$

where λ is a hyperparameter to balance discrimination and adaptation. The constraint $W^T W = I_{d'}$ enforces full

rank of matrix W.

Discussion Our framework is general and can be simplified in some cases. If only one source domain is available, then $J_c(W) = 0$ and only marginal adaptation between the source and the target is performed. We come back to the traditional domain adaptation framework seen in the previous chapter, with adaptation guarantee of the Maximum Mean Discrepancy seen in Theorem 8. As in our framework, the adaptation works only when the difference between labelling functions is small. If it is not the case, some labeled data is necessary in the target domain to use them in the conditional adaptation.

If the target domain is not specified and not available at training time, then $J_a(W) = 0$. In this case, we fall back to the framework of domain generalization. We saw in Section 2.4.1 that one can obtain guarantee on the risk of generalizing to a new unseen domains, when the number of domains becomes large [22] [125].

Our model has a few hyperparameters:

- λ: the parameter balancing both criterias
- γ: the gaussian kernel parameter
- d': the size of the learnt subspace
- K source domains

The last one is not directly a parameter but as we do not learn weights or select sources inside our model, a priori source selection is necessary to avoid negative transfer. We propose solutions to select those hyperparameters.

5.4.3 Related works

In [43], authors propose Kernel Alignment for learning a combination of kernels. Our method is different as the parameter W acts directly on the signals. Our method is in line with other linear dimensionality reduction methods on a manifold.

In [83], authors propose to learn a subspace for SPD Matrices. They formulate an objective of the form $L(W) = \sum_{i,j=1}^{n} a_{ij}\delta^2(W^T C_i W, W^T C_j W)$ where W is the parameter to learn and a_{ij} are discriminative coefficients of the form $a_{ij} = 1$ if X_i is among the K-neighbours (in the original space) of X_j of the same class, $a_{ij} = -1$ if X_i is among the K-neighbours of X_j of a different class and 0 otherwise. Their method does not make use of kernel but already gives good results. Mainly, we use the same optimization procedure as their method.

In [93], authors also propose a kernel alignment algorithm for SPD Matrices. The main difference with our algorithm is on the constraint on W. Instead of the full rank constraint, they impose W to be SPD. Unfortunately, we could not reproduce the results of their algorithm as it is unclear how they perform optimization on the space of SPD Matrices endowed with the Log-Euclidean Metric.

In [15], authors propose a dimensionality reduction for domain adaptation using MMD. For two domains and samples $X_S \in \mathbb{R}^{d \times n_S}$, $X_T \in \mathbb{R}^{d \times n_T}$, they propose to minimize $MMD(W^T X_S, W^T X_T)$. Their method is unsupervised and only minimizes difference between marginal distribution and the discriminative is done in a second stage.

Our method enhances on those two works by combining discriminative learning and adaptation. It was proved to be state of the art for domain adaptation with for instance Adaptation Regularization [114]. Other dimensionality reduction based on MMD such as TCAdo not learn a new subspace on original data but learn a kernel in the form of $k(x_1, x_2) =< w^T \phi(x_1), w_T \phi(x_2) >$ on the source data but learn a kernel, which is less interpretable.

5.4.4 Algorithm

Grassmannian optimization

To solve the objective of Problem 5.21, we have a convex cost with a full rank constraint. In fact, the problem can be formulated as an optimization over a manifold. Indeed, the set $\{W \in \mathbb{R}^{d \times d'}\}$ is also called the Stiefel Manifold. Moreover, we notice that for any $W \in \mathbb{R}^{d \times d'}$ and rotation matrix $R \in \mathcal{O}(d')$, $J(W) = J(WR)$ as the Log-Euclidean Kernel is invariant to rotation. Hence, the minimization can be done over the Grassmannian Manifold $\mathcal{G}(d', d)$, which is defined as the space of d'-dimensional linear subspace of $\mathbb{R}^d$. Classical optimization techniques such as Newton Descent or Conjugate Gradient Descent have been extended to optimization over a manifold [1].

To solve our problem, we use conjugate gradient descent. It is done in four steps:

1. Compute the euclidean gradient of the objective function $D_W J(W)$
2. Compute the Grassmannian gradient $\nabla_W J(W)$ using $\nabla_W J(W) = (I_d - WW^T) D_W J(W)$
3. Find the search direction $\mathbb{H}$ using the previous search direction and $\nabla_W J(W)$
4. Perform a line search along the geodesics of W in the direction of $\mathbf{H}$ to find the step size

The last three steps were defined in [1] and are available in the *Pymanopt* package [173], which is available in Python. The only remaining step is the computation of the gradient of $J(W)$, which involves gradients of matrix logarithms.

In [83], authors propose to approximate $logm(W^T X W)$ by $W^T logm(X) W$ using Taylor expansion, which circumvents the computations of gradients of the Log-Euclidean Metric. In fact, it is possible to compute them completely. Indeed:

$$
\begin{aligned}
D_W J(W) &= D_W \left[Tr(\mathcal{K}[W](L - YY^T) \right] \\
&= D_W \left[\sum_{i,j=1}^{Rn+m} \mathcal{K}_i j[W](l_i j - y_i y_j) \right] \\
&= \sum_{i,j=1}^{Rn+m} (l_i j - y_i y_j) D_W \left[\mathcal{K}_i j[W] \right]
\end{aligned}
\tag{5.22}
$$

Hence, we need to compute $D_W \left[\mathcal{K}_i j[W] \right]$. We detail the computation of this gradient by noting C_i and C_j the covariance matrix corresponding to samples i and j. Moreover, this computation is based on the work of [93].

$$
\begin{aligned}
D_W \left[\mathcal{K}_i j[W] \right] &= D_W \left[\| logm(W^T C_i W) - logm(W^T C_j W) \| \right] \\
&= D_W \left[Tr(logm^2(W^T C_i W) + Tr(logm^2(W^T C_j W) - 2Tr(logm(W^T C_i W) logm(W^T C_j W)) \right]
\end{aligned}
\tag{5.23}
$$

Using [32], one can compute each of those three terms

$$
\begin{aligned}
D_W(Tr(logm^2(W^T C_i W)) &= 4 C_i W Dlogm(W^T C_i W) \left[logm(W^T C_i W) \right] \\
D_W(Tr(logm^2(W^T C_j W)) &= 4 C_j W Dlogm(W^T C_j W) \left[logm(W^T C_j W) \right] \\
D_W(Tr(logm(W^T C_i W) logm(W^T C_j W))) &= 2 C_i W Dlogm(W^T C_i W) \left[logm(W^T C_j W) \right] \\
&\quad + 2 C_j W Dlogm(W^T C_j W) \left[logm(W^T C_i W) \right]
\end{aligned}
\tag{5.24}
$$

Combining those three equations, we have

$$
\begin{aligned}
D_W \left[k_i j[W] \right] = -4 (& C_i W D_{logm(W^T C_i W)} \left[logm(W^T C_i C) - logm(W^T C_j W) \right] \\
& + C_j W D_{logm(W^T C_j W)} \left[logm(W^T C_j C) - logm(W^T C_i W) \right]) \gamma \mathcal{K}_{ij}(W)
\end{aligned}
\tag{5.25}
$$

Finally, the quantity $Dlogm X[Y]$ for $X, Y \in \mathbb{S}(d')$, corresponding to the directional derivative of $logm$ at X along Y can be computed using Theorem 4.11 from [5] which states that

$$
logm \begin{pmatrix} X & Y \\ 0 & X \end{pmatrix} = \begin{pmatrix} logm(X) & Dlogm(X)[Y] \\ 0 & logm(X) \end{pmatrix}
\tag{5.26}
$$

Using the previous formula, it is possible to compute the derivative in Equation 5.25 which can be fed to the derivative of the cost (Equation 5.22).

Each gradient step of our algorithm has a cost of $\mathcal{O}((Rn + m)^2) dd'^2$, which gives a total cost of $\mathcal{O}(n_{iter}(Rn +$

$m)^2 dd'^2$. Even though with RCG, the number of iterations is small, it can be prohibitive for some applications. It is also possible to use the Stein divergence introduced in Section 5.3 instead of the Log-Euclidean Metric as it is also invariant to rotation. The main drawback is that the associated kernel stays positive definite only for some values of γ. As the choice of hyperparameter is important to the performance of our algorithm, we prefer to use the Log-Euclidean Metric.

Sparsity Constraint

As we wanted our model to be as interpretable as possible, we wanted to impose a sparsity constraint on W, in the form of

$$\begin{aligned} \min_{W \in \mathbb{R}^{d \times d'}} & J_{sparse}(W) = Tr(\mathcal{K}[W](\lambda L - Y^T Y)) + \mu \|W\|_1 \\ & \text{such that } W^T W = I_{d'} \end{aligned} \tag{5.27}$$

With such constraint, only a few original sensors would contribute to each created dimension in W, meaning that we could extract meaningful relationships between sensors. However, manifold optimization with sparsity constraint is a hard task as the ℓ^1-norm is not differentiable everywhere.

In [110], authors proposed to use subgradients ($sign(W)$ in the case of $\|W\|_1$) and prove a convergence rate in $O(k^{-1/4})$ for optimization over the Stiefel manifold. We tried to use subgradients in our algorithm to solve Problem 5.27 but the obtained solution was never sparse, even for high values of μ.

In [41], authors propose to extend proximal methods to manifold optimization. In general, proximal operators decompose the problem into two steps, one optimizing the differentiable part of the cost, and one projecting the obtained solution using a proximal operator. However, for this method to be useful, we need to be able to solve the proximal problem easily (in euclidean optimization, the proximal operator for the ℓ^1-norm has a closed form solution). For our problem, we did not find any "easy" solution to the proximal problems, but we think it is an interesting direction of work in the future.

5.4.5 Hyperparameter Selection

Dimensionality selection

We propose a method to chose the dimensionality d' based on the subspace disagreement measure introduced in [76]. Our method is based on the principal angles between subspaces. We first give ourselves a maximum

dimensionality d'_{max}. Then given K source domains, we propose to solve K subproblems as

$$W_k = \underset{W \in \mathbb{R}^{d'_{max}}}{\arg\min}\ Tr(K_k, -Y_k^T Y_k) \tag{5.28}$$

where K_k and Y_k corresponds to the submatrix of K and Y with indices of domain r. Each W_k spans a different basis. We use the notion of principal angles between subspaces to study the agreement between every basis of dimension $\leq d'_{max}$ by each W_k. For two domains k and l, we introduce $V_l \in \mathbb{R}^{d \times (d - d'_{max})}$ and $V_k \in \mathbb{R}^{d \times (d - d'_{max})}$ the orthogonal complements of W_k and W_l, *ie* $V_k^T W_k = 0$ and $V_l^T W_l = 0$. Then one can show that in the Singular Value Decomposition of $V_l^T W_k$

$$V_l^T W_k = U \Sigma V^T \tag{5.29}$$

the diagonal elements of Σ corresponds to $sin(\theta_1^{kl}), ..., sin(\theta_{d'_{max}}^{kl})$ where $0 \leq \theta_1^{kl} \leq ... \leq \theta_{d'_{max}}^{kl} \leq \frac{\pi}{2}$ are called the principal angles between W_l and W_l: in particular, for $d' \in \{1, ..., d'_{max}\}$, if $\theta_{kl}^{d'} = \frac{\pi}{2}$, then the domains are orthogonal. We want to avoid the domains to be orthogonal but still want to keep a reasonable dimensionality to build a classifier. Our criteria is formulated as

$$\min_{d' \in \{1, ..., d'_{max}\}} \{d' | \frac{1}{K(K-1)} \sum_{r \neq s} sin(\theta_{d'}^{kl}) \geq \eta\} \tag{5.30}$$

where η is a hyperparameter chosen by the user, set to 0.9 in our experiments.

Domain selection

As discussed above, our method assumes equal relationship between every source domain and the target domain. Hence, we propose a simple method to remove outlier source domains and mitigate negative transfer. Given K source domains, we propose to learn a matrix W solving

$$W^* = \underset{W \in \mathbb{R}^{d' \times d}}{\arg\min}\ Tr(\mathcal{K}_S[W], -Y_S^T Y_S) \tag{5.31}$$

where $\mathcal{K}_S$ and Y_S corresponds to the submatrix of K and Y with indices of every source domain. W^* is a matrix that learns how to classify every domain. Then we compute the error made by W^* on each domain $\epsilon_k(W^*)$. We remove every domain such that $\epsilon_k(W^*) \leq \eta_d$ where η_d is a parameter set by the user. In our experiments, we set it to the mean of the errors over every domain minus two standard deviations.

Leave-one-domain-out CV

Cross-validation is a popular method for hyperparameter selection in traditional Machine Learning. However, in unsupervised domain adaptation, we do not have labels in the target domain, hence, traditional cross-validation

cannot be used. We propose leave-one-domain-out cross-validation to select γ and λ. First, we define a grid for possible values of γ and λ. Then, for every set of hyperparameters, we iteratively learn our model on $K-1$ domains with the remaining domain used as a target domain. The chosen hyperparameters are the ones giving the overall lowest error rate. Note that we could use this scheme to select the dimensionality as well but it requires to learn our model for every possible set of hyperparameters. Our dimensionality selection scheme is less costly and leads to good results in practice.

5.5 Numerical Results

We present the results of our method on both synthetic and real world data. We first simulated synthetic data using a Vector AutoRegressive model, which is often used to analyze industrial multivariate time series. Then we experimented on Human Activity Recognition datasets: UCI Daily Activites [9] and NTU RGB-D 120 Dataset [163]. The next step is to apply our method on industrial dataset at EDF when they are collected and pre-processed.

5.5.1 Simulated data

We use two procedures to simulate synthetic data: in a first example, we directly simulate covariance matrices corresponding perfectly to our framework. In a second example, we simulate source and target domains using a VAR model, which should be harder for our method but closer to real world data.

Covariance data

We first simulate synthetic data corresponding to the assumptions of our models. We generate a binary classification ($\{+1,-1\}$) problem over 10 domains by generating two SPD covariance matrices with dimension $d=20$, noted $C_k^{(+1)} \in \mathbb{S}(20)$ and $C_k^{(-1)} \in \mathbb{S}(20)$ on each domain k. Then, we generate $n=100$ samples $(x_k^{(i)}, y_k^{(i)})$ such that $x_k^{(i)} \in \mathbb{R}^{20,1000}$, $y_k^{(i)} \in \{-1,+1\}$ and $x_k^{(i)}[t] \overset{\text{i.i.d}}{\sim} \mathcal{N}(0, C_k(y_k^{(i)})) + \epsilon_t$ where $\epsilon_t \sim \mathcal{N}(0, I_d)$. The goal is to classify each time series $x_k^{(i)}$ to its label $y_k^{(i)}$.

The covariance matrices $C_k^{(+1)}$ and $C_k^{(-1)}$ are different in each domain but we assume that every domain share the same relationships on the first $d'=4$ dimensions: $[C_1^{(+1)}]_{1:4} = ... = [C_{10}^{(+1)}]_{1:4}]$ and $[C_1^{(-1)}]_{1:4} = ... = [C_{10}^{(-1)}]_{1:4}]$.

In our evaluation, we consider $R=9$ domains as the source domains and the last one as the target domain. We estimate the covariance from each $x_k^{(i)}$ and use them as features in five different methods:

1. **AIM-MDM:** Minimum Distance to Mean classifier with the Affine Invariant Metric without Dimensionality Reduction

2. **LR-MTL:** multi-task learning with a linear classifier with group-sparsity constraint presented in [10]

3. **LEK-SVM:** Support Vector Machine classifier with the Log-Euclidean Kernel without Dimensionality Reduction
4. **LEK-DR-SVM:** Support Vector Machine classifier with the Log-Euclidean Kernel with Dimensionality Reduction using only discriminative learning (J_y) and no transfer
5. **LEK-DG-SVM:** Support Vector Machine classifier with the Log-Euclidean Kernel with Dimensionality Reduction using only discriminative learning (J_y) and conditional adaptation (no information from the target domain is used)
6. **LEK-TL-SVM:** Support Vector Machine classifier with the Log-Euclidean Kernel with Dimensionality Reduction using our model in Equation 5.21

The last three methods are based on our model: **LEK-DR-SVM** does not perform adaptation and is similar to [93], **LEK-DG-SVM** corresponds to domain generalization and **LEK-TL-SVM** corresponds to domain adaptation. In **LR-MTL**, the specific geometry of the covariance matrix is ignored, as it takes vectors as an input. We set the dimensionality $d' = 4$, keep every domain and use leave-one-domain-out cross validation to select γ and λ for the last four methods. We report the average accuracy over every domain in Table 5.1. We also report the result of intra-domain learning using **LEK-SVM** learned on the target domain and tested on the same domain.

Method	AIM-MDM	LR-MTL	LEK-SVM	LEK-DR-SVM	LEK-DG-SVM	LEK-TL-SVM	Intra-domain
Accuracy (%)	56.12	72.87	55.41	53.28	**98.79**	97.52	100

Table 5.1: Average accuracy for synthetic covariance data

One can see that when learning and testing on the same domain, the classification is done perfectly, with accuracy of 100%. It does not hold in the cross-domain scenario, both with Affine Invariant Metric and Log-Euclidean Metric. The dimensionality reduction does not give good results either without adaptation. **LR-MTL** improves on those methods, but fails to find relevant dimensions for some domains. **LEK-DG-SVM** and **LEK-TL-SVM** both give very good results, almost as good as the intra-domain scenario. In Figure 5.2, we also plot the obtained matrix W for one experiment. One can see that only the common variables between every domain are selected. Previous results demonstrate the ability ouf our method. We still underline that the generated problem is very simple and has been designed perfectly for our method. Hence, we also propose to simulate synthetic data using a VAR model.

Vector AutoRegressive model

We also simulated a similar dataset using the Vector AutoRegressive model. We consider $R = 10$ domains, time series with $d = 20$ sensors and a binary classification problem with $y \in \{-1, +1\}$. For each domain and each class, we generate random VAR coefficients with horizon 5 (VAR(5) model): $A_1^{(r,-1)}, ..., A_5^{(r,-1)}, A_1^{(r,+1)}, ..., A_5^{(r,+1)} \in \mathbb{R}^{20\times 20}$. To enforce stationarity of the VAR model, we constrain the coefficient matrices to have singular values lower than 1. Then, we constrain the model parameters to be the same for the first 4 dimensions in every domain for each

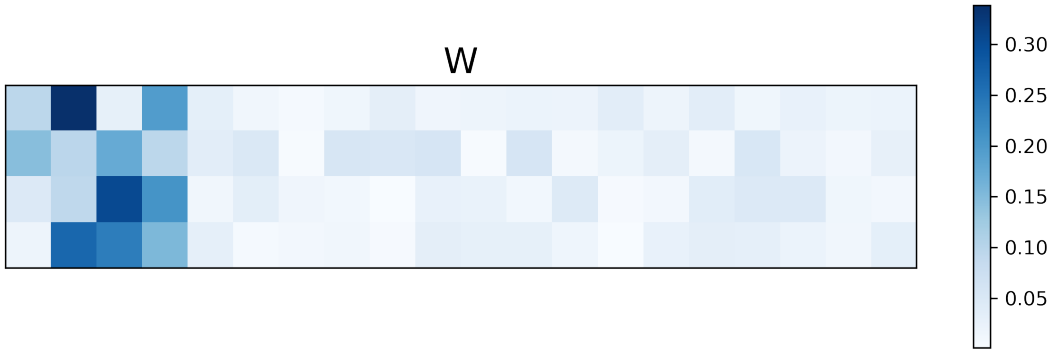


Figure 5.2: Parameter W learned by the model for the synthetic covariance experiment ($abs(W)^T$ is represented here and values are normalized to sum to 1 for each dimension for better readability)

class, *ie* $[A_l^{(1,-1)}]_{1:4} = ... = [A_l^{(10,-1)}]_{1:4}$ and $[A_l^{(1,+1)}]_{1:4} = ... = [A_l^{(10,+1)}]_{1:4}$ for $l = 1, ..., 5$ and where $[A_l^{(r,y)}]_{1:4}$ is the square sub-matrix of $A_l^{(r,y)}$ for indices lower than 4.

Finally, we generate $n = 100$ samples $(x_k^{(i)}, y_k^{(i)})$ such that $y_k^{(i)} \in \{-1, +1\}$, $x_k^{(i)}$ has a length 1000 and

$$x_k^{(i)}[t] = \sum_{l=1}^{5} A_l^{(r,y_k^{(i)})} x_k^{(i)}[t-l] \tag{5.32}$$

In our evaluation, we consider $R = 9$ domains as the source domains and the last one as the target domain. We estimate the autocovariance matrix with horizon 5 from each $x_k^{(i)}$. Note that the horizon was chosen by hand, but could be chosen from the traditional Partial Autocorrelation Function method. We tested the five methods described above and report the average accuracy over every domain in Table 5.2.

Method	**AIM-MDM**	**LR-MTL**	**LEK-SVM**	**LEK-DR-SVM**	**LEK-DG-SVM**	**LEK-TL-SVM**	**Intra-domain**
Accuracy (%)	53.28	42.12	57.87	59.72	**92.28**	92.19	100

Table 5.2: Average accuracy for synthetic VAR coefficients

We make similar observations to the previous experiment, which can be explained by the fact that both metrics are invariant to inversion. Once again, the methods that extract transferable features are the ones using conditional adaptation and there is no gain from the marginal adaptation using target data. One can note that **LR-MTL** performs worse than a random classifier as it does not take the specific geometry of covariance matrices. On the other hand, when we directly feed AR coefficients to **LR-MTL**, it performs as well as in the previous experiment.

As a conclusion, our method allows to find relevant dimensions for domain generalization and transfer learning for synthetic data, both for covariance and VAR models. While the assumptions made by our model and experiments rarely are perfectly met in practice, we experiment on real world datasets in the next section.

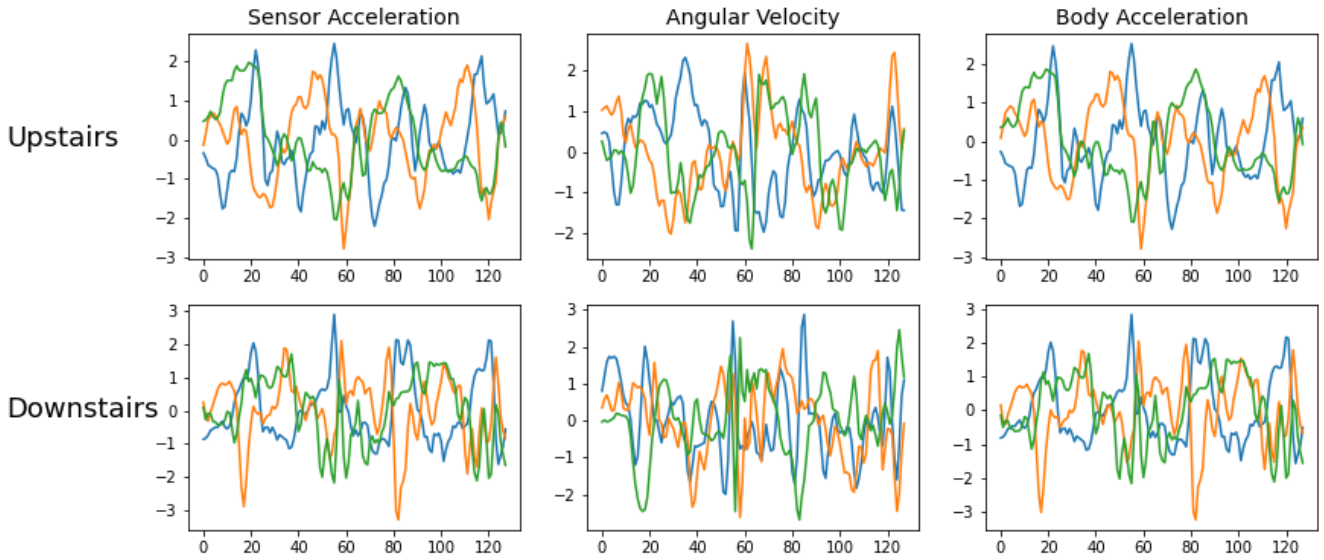


Figure 5.3: Examples of time series from UCI Daily Activities Dataset: top row represents walking upstairs and bottom row corresponds to walking downstairs. On each figure, blue is the x-dimension, orange the y-dimension and green the z-dimension.

5.5.2 Human Activity Recognition

We evaluate our method on two Human Activity Recognition (HAR) datasets: UCI Human Activity Recognition dataset [9] and NTU RGB-D [163] dataset. It could be useful in EDF production facilities for worker safety monitoring especially when workers have to work alone, such as in a power plant. Moreover, it illustrates how our algorithm can perform on a domain adaptation task for multivariate time series.

UCI Daily Activities dataset

This public dataset is made of motion sensor data of 6 daily and sports activities performed by 30 subjects for 2.56 seconds. Each subject is monitored with a smartphone worn on the waist. Using the accelerometer and the gyroscope of the smartphone, each activity corresponds to a 9-dimensional time series: 3-dimensional acceleration, 3-dimensional triangular velocity and 3-dimensional estimated body acceleration. Every sensor is sampled at $50Hz$, giving a time series of length 128. In total, 10299 samples are available over the 30 subjects. In Figure 5.3, we illustrate the monitored time series for the *walking upstairs* and *walking downstairs* classes.

Here, we only represented each time series with its covariance matrix $C \in \mathbb{S}(9)$ as results were not better when using autocovariance matrix. Moreover, the time series length is 128 which makes the estimation of autocovariance matrix unstable. We used ledoit-wolfe regularization to estimate the covariance matrices.

Our dimensionality selection method gave a subspace size of $d' = 3$. We report the average accuracy of the 5 methods in Table 5.3. Our method **LEK-TL-SVM** outperforms every other method that does not use adaptation although results obtained from the **LEK-SVM** are already good.

Method	AIM-MDM	LR-MTL	LEK-SVM	LEK-DR-SVM	LEK-DG-SVM	LEK-TL-SVM	Intra-domain
Accuracy (%)	77.21	78.12	88.55	87.28	93.56	**93.72**	100

Table 5.3: Average accuracy for Human Activity recognition with smartphones dataset

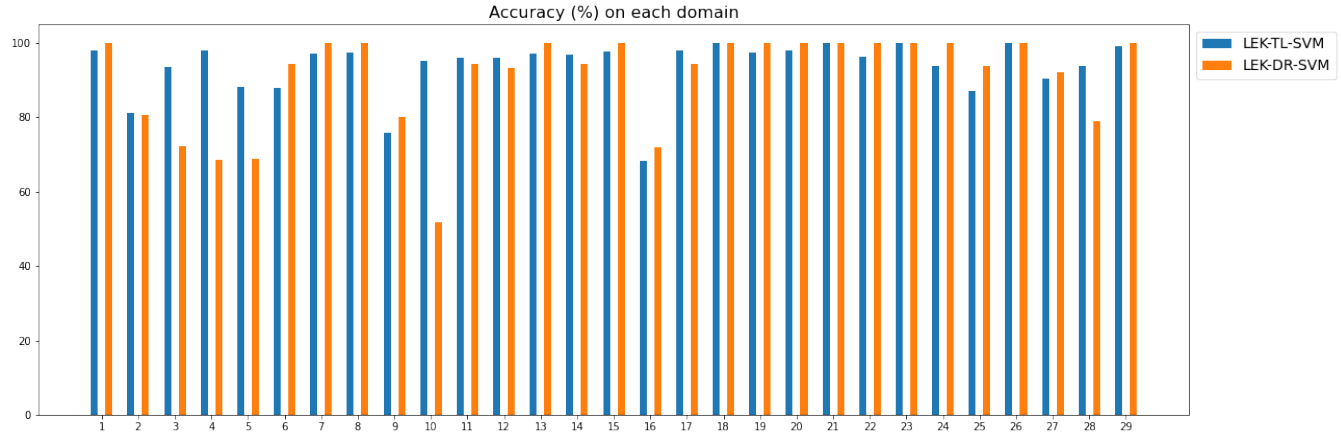


Figure 5.4: Accuracy (%) of **LEK-SVM** (orange) and **LEK-TL-SVM** (blue) over every domain

Our method helps to increase the performance on several domains on which the **LEK-SVM** underperformed. To visualize it, we represent on Figure 5.4 the accuracy of the **LEK-SVM** (without adaptation) and **LEK-TL-SVM** (with adaptation) on every domain. One can see that the lowest accuracy obtained without adaptation is close to 50%, whereas with our adaptation scheme, the lowest accuracy is around 70%. Overall, we noticed that our method allows to extract features that are more robust to a domain change.

The matrix W learnt by our model indicates that the most relevant features are the z-acceleration and the y-estimated body acceleration when the last dimension is a mixture of y-rotation and z-estimated body acceleration. We represent the obtained matrix in Figure 5.5.

Finally, it should be noted that better results can be obtained with deep learning methods, even without adaptation. When we use a CNN by learning on 29 subjects and applying on the last one, we obtain overall accuracy of 96.14%, even without adaptation. However, our method is more interpretable as the weights can be visualized and is shorter to train (less than 10 minutes as the optimization converged in 5 iterations).

NTU RGB-D Dataset

NTU RGB-D[1] [163] is a dataset collected at NTU where 40 subjects were filmed while performing 60 actions. The 60 actions are various actions of everyday life, such as drinking water or running.

While the dataset can be considered as a video dataset, we use the 3D skeleton data: for each video, the 3D-coordinates of 25 body joints (see Figure 5.6) are collected at each frame. Hence, we consider as an input time series of dimension 75. The duration of the sequence is different for every video: it is not an issue with our method

[1] It was later extended to NTU RGB-D 120 dataset

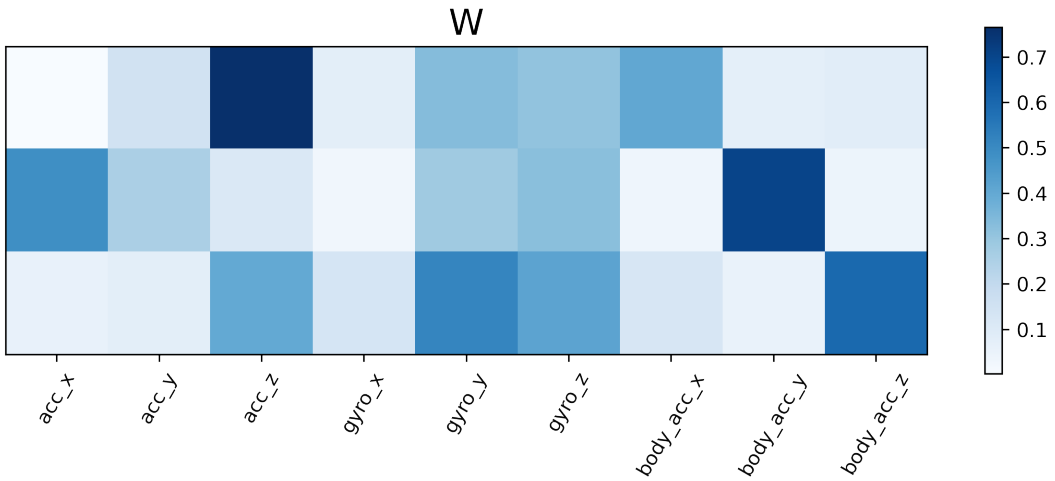


Figure 5.5: Parameter W learned by the model for the HAR with smartphones experiment ($abs(W)^T$ is represented here and values are normalized to sum to 1 for each dimension for better readability)

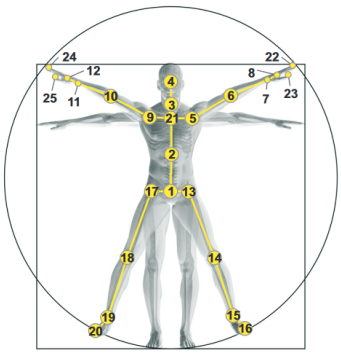


Figure 5.6: Positions of the 25 body joints monitored with sensors. Figure is taken from [163]

as we only use covariance information, which can be compared between two time series of different sizes. Once again, the length of the time series is very short (~ 100), hence we only use covariance information. We did not use autocovariance, as they did not improve the results and led to even bigger SPD Matrices whose estimation is more unstable.

In total, $60,000$ samples are available. For this dataset, state-of-the art results were obtained using recurrent neural networks but covariance-based methods were also successful [38] [182] [62]. Those methods did not take into account the specific geometry of SPD Matrices but focused on creating hand-engineered features for skeleton-based activity recognition.

Whole dataset In our experiment, we iteratively take 39 subjects as source domains and the last one as the test domain. Once again, we tried the 5 methods described in the previous experiments. We also tried to use a Temporal Convolutional Network (**TCN**) with three causal convolution layers, with size 128, 128 and 256 and a dilation rate of 2 at each layer, with a dense layer in the end. The only difference with univariate time series is that the input

data has 75 channels (dimensions) instead of 1. Better results could probably be obtained with better architecture search but we simply wanted to compare with our method. Finally, we also tried to use our scheme proposed in Chapter 4 **AMSHDM**: we used the TCN as the feature extractor and a simple dense layer as the final classification layer.

For our method **LEK-TL-DR**, the dimensionality selection gave values between 6 and 9 in the experiments (for different domains). For better readibility of the results, we set $d' = 8$ for every domain. The domain selection often removed two of the subjects that are far from every other. We give the average accuracy over every subjects in Table 5.4. Moreover, we also report results obtained from other papers (we did not run their experiments again and directly report results from their paper). Note that their framework for the cross-subject evaluation is different from ours: they split the dataset in two sets of 20 training subjects and 20 test subjects. We also report the runtime for our methods.

Method	Accuracy (%)	Run-time
Covariance-based		
AIM-MDM	30.58	22h (CPU)
LR-MTL	41.32	11m (CPU)
LEK-SVM	36.78	6h (CPU)
LEK-DR-SVM	37.23	14h (CPU)
LEK-DG-SVM	57.78	17h (CPU)
LEK-TL-SVM	**58.51**	16h (CPU)
CovP3DJ [62]	51.40	?
Hand-crafted features		
HOG^2 [136]	32.24	?
Super Normal Vectors [189]	31.82	?
Lie Group [178]	50.08	?
FTP Dynamic Skeletons [89]	**60.23**	?
Deep-learning based		
P-LSTM [163]	62.93	?
TCN	74.78	54m (GPU)
AMSHDM	78.4	1h32m (GPU)
MS-AAGCN [164]	**90.00**	?

Table 5.4: Average accuracy and run-time for some methods on the NTU RBG-D dataset

A few observations can be made from these results. Firstly, similarly to other experiments, our method outperforms every other covariance-based method in the transfer learning scenario. Once again, the improvement obtained with conditional adaptation is important but the marginal adaptation to the target domain brings only a small improvement. Moreover, our method outperforms **CovP3DJ** [62], which created specific features on which to compute the covariances. Method with hand-crafted features give lower overall accuracy, apart from **FTP Dynamic Skeletons** [89]. On the other hand, every deep learning method gives better results, from a basic LSTM (**P-LSTM** [163]) to the **TCN** we used. Note that the current state-of-the art on the dataset is **MS-AAGCN** [164] which is an attention based LSTM where the architecture is specific to skeleton data. For every deep learning method, every source domain is simply concatenated to create a training set and no transfer learning is used.

The run-time is clearly an issue for our methods with such dimensions as there are $60,000$ covariance matrices of dimension 75×75. Indeed, computing the kernel and computing each gradient step is very long. It is even worse when using the Affine Invariant Metric for which the pairwise distances are longer to compute. The proposed **TCN** and **AMSHDM** are shorter to train, as we used Pytorch with a GPU-backend on a clusters of two GPUs. One could still note that (i) when using a CPU-backend, the networks took more than two days to be trained (ii) the computation of the kernel matrix and the gradient step could be parallelized, which would speed up the computations. Moreover, at test time on a new subject, if we directly use the learned W (domain generalization **LEK-DG-SVM**) our method is very quick as the kernel matrix can be computed quickly (given that the matrix-logarithm of every training sample is cached in memory and the new dimensionality is small).

Overall, when using the whole dataset, there is enough diversity in the data for neural network-based methods to perform well. Moreover, our method does not scale very well and the run-time becomes an issue. On the other hand, our method already gives better results than other covariance-based methods including the ones that creates features specific to Human Activity Recognition. To better illustrate our method, we use a smaller dataset in an additional experiment.

Small dataset In a second experiment, we use a subset of the NTU RGB-D dataset. We select only the first 10 subjects and 2 actions (classes): drink water and clapping. The new dataset has a total number of 631 frames.

We use the same scheme as before, by iteratively selection 9 subjects as source domains and the remaining one as target. The dimensionality selection gave values between 5 and 7 and we kept $d' = 5$ for every experiment. We give the average accuracy over every subjects in Table 5.5.

Method	Accuracy (%)	Run-time
Covariance-based		
AIM-MDM	74.21	8m (CPU)
LR-MTL	70.18	1m (CPU)
LEK-SVM	82.37	2m (CPU)
LEK-DR-SVM	84.57	6m (CPU)
LEK-DG-SVM	88.77	6m (CPU)
LEK-TL-SVM	**89.92**	6m (CPU)
Deep-learning based		
TCN	**91.21**	1m (GPU)
AMSHDM	87.31	1m (GPU)

Table 5.5: Average accuracy and run-time for some methods on the reduced NTU RBG-D dataset

Once again, our method gives better results than other covariance-based methods. Deep Learning still performs better but the gap is not as big as on the whole dataset. The main interest of our method lies in its interpretability: indeed, we are able to identify which features contribute the most to each new dimension. We saw that the accelerations at joints positioned on the fingers, arms and head were given the most weight, as represented on Figure 5.7. It makes sense as for clapping and drinking water, joints on the lower part of the body do not have huge influence

and only add noise.

It can also be seen on Figure 5.8 where we represented the Kernel PCA of the log-euclidean kernel on covariances computed on the original sensors and on covariances computed on the reduced space. One can see that with our method, classes are well separated and domains are mixed together. On the other hand on the original data, the noise added by the uninformative features makes it harder to separate classes.

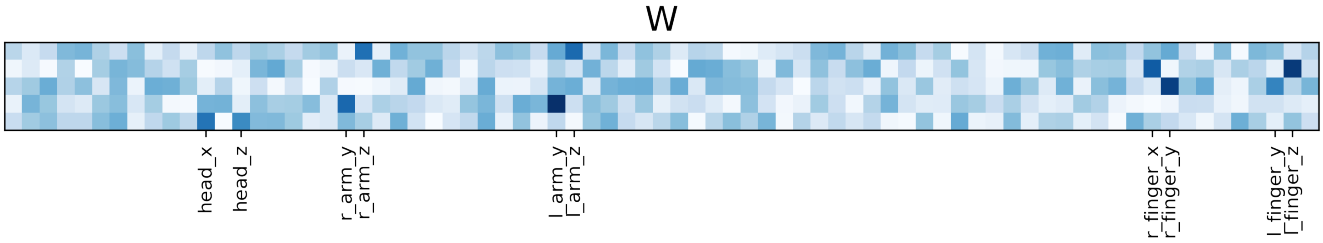


Figure 5.7: Parameter W learned by the model for the reduced NTU RGB-d dataset ($abs(W)^T$ is represented here and values are normalized to sum to 1 for each dimension for better readability)

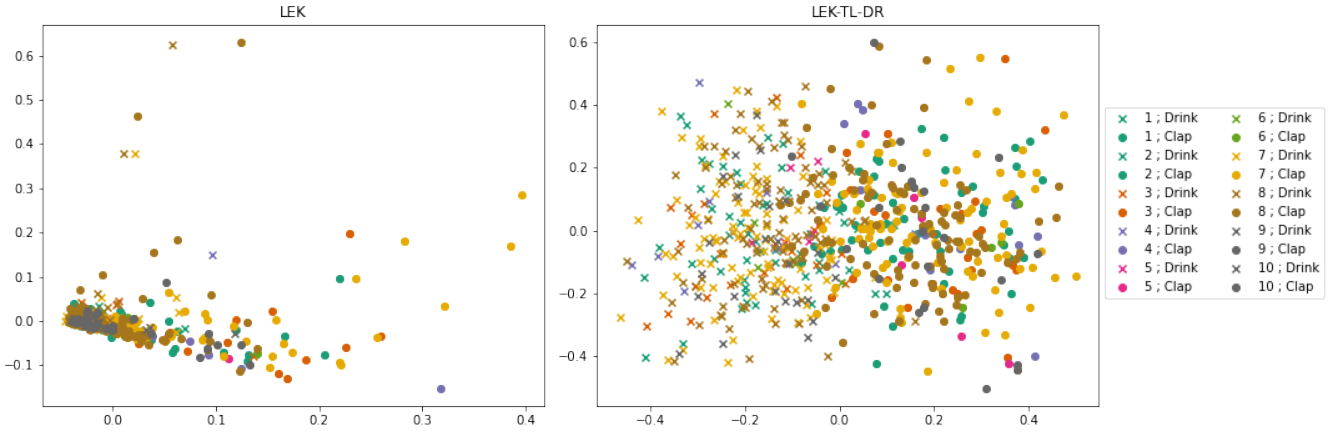


Figure 5.8: Kernel PCA representation of the Log-Euclidean Kernel on the original data (left) and after dimensionality reduction (right). Each colour corresponds to a domain and classes are represented with two markers.

Conclusion

With these experiments on real-world datasets, we can draw several conclusions on the advantages and the limits of our method. Mainly, we showed that our method outperforms other covariance based methods in a transfer learning scenario, as it is the first method to combine discriminative learning and adaptation based on covariance information. For Human Activity Recognition, it can be sufficient to get good performance and interpretable features. On the other hand, better performance can be obtained with deep learning methods, especially when the training set is large. In this case, neural networks are able to learn features robust to a change of subject. When the training set is smaller, our method gets similar performance to deep learning methods with more interpretability.

5.6 Conclusion

This Chapter proposed a method for transfer learning based on covariance data. This is the first method that proposes to learn a linear dimensionality reduction on time series data with two objectives: discriminative learning and transfer learning. We showed in our experiments the applicability of our methods to both synthetic multivariate time series and human activity recognition. While deep learning methods provide better performance when the learning set is large, our method extracts interpretable features and the performance gap becomes smaller when the dataset size is small.

Our work opens many perspectives for covariance-based methods for time series. Firstly, we saw that the runtime of our method suffers when the training set becomes too large. While it could be mitigated with a parallelized implementation of our algorithm, methods that are not based on kernels could also be used. For instance, a new line of work tried to design deep neural networks for SPD Matrices. This means that the methods developed in Chapter 4 could be extended to SPD Matrices.

Secondly, we led experiments on synthetic data created with a Vector AutoRegressive model representative of an industrial system. We would like to try our method on real industrial data, but at the time of this work, most datasets available at EDF were outlier detection problems and did not contain labels. Our method could be used for outlier detection by modifying the discriminative learning objective to an unsupervised dimensionality reduction as in [83], but mixing the two objectives of matching distributions across domains and separating outliers would need a detailed analysis.

As discussed before, we also would like to add a sparsity constraint to our algorithm to obtain more interpretable dimensions. This is directly linked with manifold optimization and we think a solution based on proximal methods similar to [41] could be obtained for our problem.

Finally, we used the specific geometry of covariance and auto-covariance matrices as Symmetric Positive Definite matrices. When working on time series, we know that the temporal dependence is important, which is captured by auto-covariance matrices. Auto-covariance matrices are block-Toeplitz when working with stationary time series, and block-Toeplitz matrices also have a specific geometry [99]. Some classical algorithms such as K-means [36] were derived using this specific geometry and it would be interesting to see if it is possible to include it in a transfer learning method.

Chapter 6

Conclusion and Perspectives

In this thesis, we proposed novel methods to extract transferable features from time series. We proposed different approaches for learning transferable features using neural networks: one using an ensemble of normalizations and the second one following adversarial learning. In our adversarial learning method, we showed that our new measure of discrepancy unifies previous works. Our framework can be applied in both classification and regression problems, and is generic to other kinds of data. In a last chapter, we developed a framework to extract more interpretable features for time series. Building on previous works on learning from covariance matrices, we proposed a method to learn invariant relationships between sensors in a multivariate time series. When a large dataset is available, deep learning methods outperforms our method, however, when the dataset size is smaller, our method leads to similar performance with better interpretability. Our methods can answer several problems of real-world applications. Firstly, for many applications such as Non Intrusive Load Monitoring, data collection is expensive. Our methods can use existing resources to learn from fewer new data. Secondly, training an algorithm requires computing power and energy. More and more methods pose the objective of learning on a budget, *ie* learning with a limited computing or environmental cost. By pre-learning an invariant subspace of features, the method proposed in Chapter 5 reduces the cost of learning on a new dataset.

Perspectives

Further experiments. For every method proposed in this thesis, we conducted extensive experiments on synthetic and real-world datasets. For applications to NILM, one could still note that the size of the training and test sets is always limited. It would be interesting to carry out similar experiments on larger datasets when they become available. Moreover, we ony tried to deal with daily consumption estimation, which would be sufficient for most applications for individual households. Industrial warehouses or production sites are also interested in energy disaggregation and it would be interesting to try our method for those larger buildings. While we conducted experiments on synthetic data

and Human Activity Recognition, the next step is to test the method of Chapter 5 on industrial problems.

Larger Training Sets. Having larger training set would also probably allow the learning of a general architecture that could be fine-tuned to a new small target dataset, in a similar way to AlexNet [103] for images. We argue that the main effort of such work would not reside on finding the best architecture but rather on the collection of the dataset: the collection of ImageNet took several years for a dedicated team to collect images, pre-process them, and label them. Collecting such dataset for individual household electricity consumption would be expensive and raises privacy issues. Assuming the existence of such a large training set, one could learn a large model. Knowledge distillation [86] can then be used to create smaller models that can be deployed on each sensor.

Data Augmentation. We saw that results obtained with transfer learning are equivalent to the ones obtained with data augmentation. Indeed using a Multi-Agent System to simulate the electricity consumption of a house, we are able to generate a larger training space and increase the generalization ability of an algorithm learnt on this new data. This is related to the topic of learning from simulations. We think that in the context of transfer learning or domain adaptation, it would be possible to drive the simulation using the target domain. It would allow to increase the knowledge on the data space around the target domain. This could be done using field-specific information or using statistical learning, as was proposed with GANs [92].

Interpretability. Interpretability is very important for the deployment and acceptance of statistical models by operators of a system. Neural Networks are very popular and state-of-the-art in many applications but are still hard to understand. A better understanding of features learnt from time series and linking them with real-world quantites would allow them to be better accepted by users or authorities. For our covariance-based method, imposing a sparsity constraint would allow to directly extract measured quantities. This requires more work on optimization over a manifold, where proximal methods such as [41] seem to be a good starting point.

Federated Learning. As the public becomes more knowledgeable about the usage of its data, privacy preserving methods are an interesting solution. In all of our experiments, we assumed that the whole dataset was available on our machines to fit our algorithm. Each household in France has or will have a smart sensor (Linky) able to directly send information to the electricity provider. Instead of uploading the whole electricity consumption to a central server held by EDF, one could imagine training an algorithm across multiple smart sensors, without storing or exchanging the raw electricity consumption. This problem is tackled by federated learning. Federated Learning already attracted attention from the transfer learning community, as different nodes most likely have different distributions [124]. Having a centered model that can be updated in a decentralized manner and adapt to each node would be ideal.

Learning causal relationships. In Chapter 5, we proposed a method to learn relationships between sensors in a multivariate time series that were invariant to a change of system or subject. While we formulated these relationships using covariance, or equivalently partial autocorrelation as our metric is invariant to inversion. While we did not formulate it in terms of causal learning, connections can be made. In causal learning, the goal is to learn the causal graph between variables that is robust to a change of environment. By exchanging environment with domains, one can see that causality and domain generalization are closely related. It was noted in a recent work [154] where authors propose a method to combine causality and transfer learning.

Heterogeneous Transfer. Methods presented in this thesis assume homegeneity in the input and prediction space across domains. In the future, it could be interesting to extend our work to heterogeneous data space. For instance, industrial systems are often monitored with a different set of sensors. With our method, one should remove the different dimensions from the dataset. Finding a way to match distributions from multivariate time series with different dimensionalities would allow to make us of even more information. For instance, [153] tried to propose such a method for EEG data. Even more ambitious would be to merge information from data of different natures: for instance, some industrial systems are monitored with sensors but operators also write reports. Merging time series data and textual annotations has been proposed for time series forecasting [135] and could be extended to a transfer learning framework.

Appendix A

Neural Networks

Basics In this Appendix, we give a background to neural networks and their training. Formally, neural networks are a function $\Phi : \mathcal{X} \to \mathcal{Y}$ from an input space $\mathcal{X}$ to a prediction space $\mathcal{Y}$. Typically, $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} = \{-1, +1\}$ for binary classification. Phi is a composition of several functions, called layers, such that if Phi is a neural network of depth L, $\Phi(x) = f_L \circ ... \circ f_1(x)$.

The most classical form for a layer l is the dense layer defined by:

$$z_l = f_l(z_{l-1}) = \sigma(w_l^T z_{l-1} + b_l) \tag{A.1}$$

where $z_l \in \mathbb{R}^{d_l}$ is the output of layer l, $w_l \in \mathbb{R}^{d_l \times d_{l-1}}$ is the weight matrix of layer l, $b_l \in \mathbb{R}$ is the bias of layer l and σ_l is an activation function. An neural network architecture made of such layers is also called fully connected as every neuron is connected to another. The prediction of an input vector x is given by

$$y = \Phi(x) = \phi_L(\phi_{L-1}(...(\phi_1(x)))) \tag{A.2}$$

The number of layers L, number of neurons at each layer d_l and activation functions σ_l are hyperparemeters of the model and chosen by the user. Typically, activation functions are non-linear. Common activation functions include

$$\begin{aligned}
\sigma(z) &= max(z, 0) && \text{(ReLu)} \\
\sigma(z) &= x * \mathbf{1}_{x \geq 0} + \alpha(e^x - 1) * \mathbf{1}_{x \leq 0} && \text{(ELu)} \\
\sigma(z) &= \frac{1}{1 + e^{-z}} && \text{(Sigmoid)} \\
\sigma(z) &= \frac{e^x - e^{-x}}{e^x - e^{-x}} && \text{(tanh)}
\end{aligned} \tag{A.3}$$

The choice of architecture is dependent on the application and there is no general rule as to which functions to choose.

Back-propagation The parameters w_l and b_l are learnt by the model using feed-forward backpropagation. The most popular way to train a neural network is using Gradient Descent. We consider a sample of size n $\{(x_1, y_1), ..., (x_n, y_n)\}$. Then given a differentiable loss function L, the objective function minimized by the neural network is

$$J(\Phi) = \sum_{i=1}^{n} L(y_i, \Phi(x_i)) \tag{A.4}$$

Typically, for classification the cross-entropy loss is used whereas for regression, the mean squared error is used. If we write $\theta = \{b_1, w_1, ..., b_L, w_L\}$ the parameters of our neural network and write $J(\Phi) = L(y_i, \Phi(x_i)) = L(y_i, x_i; \theta) = J(\theta)$, then one gradient step at iteration t done by a traditional gradient descent would correspond to

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t \sum_{i=1}^{n} \nabla_\theta L(y_i, x_i; \theta^{(t)}) \tag{A.5}$$

where η_t is the learning rate at epoch t.

In general, as the training set is large, Stochastic Gradient Descent is used. The gradient step defined above can simply be replaced by a gradient step over a small batch instead of the whole training set.

Extensions have been proposed by adding momentum using Nesterov's accelerate gradient. Additionally, the learning rate η_t can be fixed, chosen to decay or adapted during traing. In our experiments unless otherwise indicated, we always used *Adam* optimizer.

Batch-Normalization Batch-normalization [95] is a technique to improve the training of neural networks. Namely, it first normalize the output of a layer over a mini-batch: for a batch of size B at layer l, for $k \in \{1, ..., B\}$, the normalization step is done with $\widehat{z_l^{(k)}} = \frac{z_l^{(k)} - \mu_B}{\sigma_B}$ where μ_B and σ_B are the empirical mean and standard deviations of outputs z_l. As such, $\widehat{z_l^{(k)}}$ has zero mean and unit variance. However, in order not to allow the network to learn by itself the "useful" mean and variance for the given layer the output of a batch-normalization layer is given by

$$BN(z_l^{(k)}) = \gamma_l \widehat{z_l^{(k)}} + \beta^l \tag{A.6}$$

where γ_l and β_l are learnt using backpropagation. Batch-normalization is said to reduce the internal covariate shift, thus accelerating training of neural networks but understanding its effects is still an active topic.

For models in Chapter 4, we used batch-normalization. In Chapter 3, we did not use batch-normalization in our normalization ensembling model.

Dropout Dropout is a technique used to avoir overfitting with a neural network. Indeed, overfitting is a problem as often neural networks are over parameterized. It corresponds to dropping a random subset of units (neurons) during training, such that a smaller sub-network is learnt at each pass. We used dropout in every of our model as it helped to reduce the gap between training and validation error.

Convolutional layers Dense layers are the most basic layers used in a neural network. When working on structured data such as images or time series, convolutional layers allow to take into account temporal or spatial dependence. Formally, a convolutional layer is defined by a transformation :

$$z = \sigma(\sum_{j=1}^{r} W^j x_{1+(i-1)s+j} + b_j) \tag{A.7}$$

where $W \in \mathbb{R}^r$ and $b \in \mathbb{R}^r$ are respectively the kernel matrix and the bias of the layer to be learned with kernel size r and stride s. With such layers, temporal structure is kept as adjacent points contribute to the same output in the next layer.

Appendix B

Clustering consumer consumption with auto-encoders

In this Appendix, we illustrate a deep learning method to cluster consumers based on their consumption behaviour. This work has been presented in a national conference [150]. It shows the improvement of using deep learning over classical representations for time series clustering.

Our goals is to perform whole time series clustering. This was developed for applications in consumer segmentation. Indeed, if EDF can identify different customer behaviours, it is possible to offer tailored contracts. For this purpose, we propose a method to cluster EDF clients based on their daily consumption over a year. We propose to use deep convolutional autoencoders in order to extract representations on which the clustering is applied. Our experiments, conducted on real world public data, show that our method gives good results in term of robustness to outliers.

B.1 Data presentation

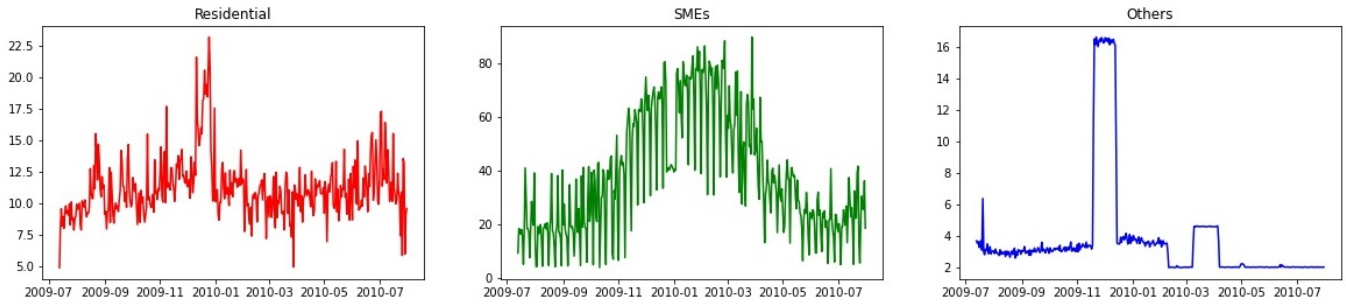


Figure B.1: CER Smart Metering data: (red) Mean of Residential consumption (green) Mean of SMEs consumption (blue) Mean of Others consumption

Clustering consumer based on their consumption has several interests for EDF: a better knowledge of its clients allows to propose tailored solutions. Moreover, some consumption forecasting methods are based on an ensemble of predictors learnt on clusters of clients.

Here, we illustrate the obtained results on a public dataset of electricity consumption in Ireland: CER Smart Metering project [69]. This experiment collected consumption data from 3,174 [1] Irish households and SMEs between 2009 and 2010. We only consider **daily consumption** for the first 384 days of the experiment. A survey on households was conducted and for some consumers, we know if they are residential households or SMEs. When left un-answered, the monitored house can be either of those classes or a secondary residence. The general pattern for SMEs is a lower consumption during week-ends and Christmas vacations, which is opposite to residential households.

B.2 Method

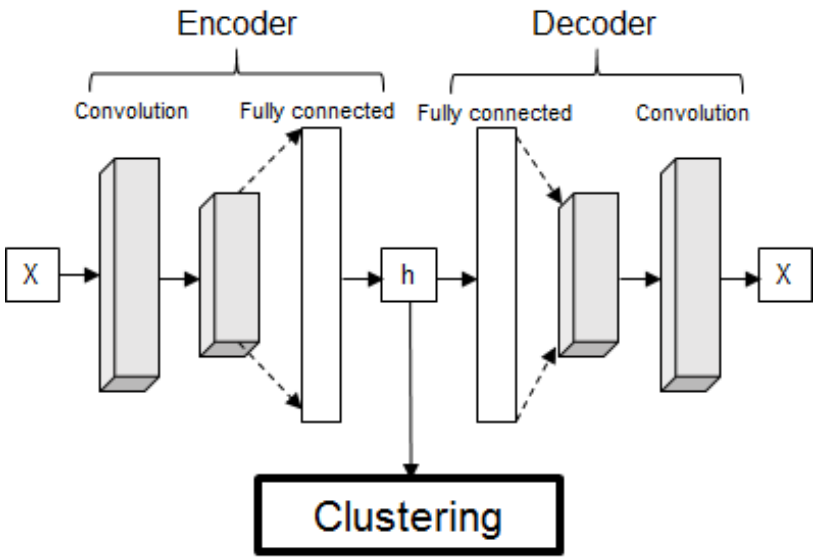


Figure B.2: Proposed method: Convolutional Auto-Encoder (CAE)

B.2.1 Convolutional AutoEncoder

Our goal is to cluster the clients, hence we use an unsupervised architecture with a convolutional autoencoder (CAE). The CAE is made of an encoder, which follows the general convolutional architecture, a bottleneck and a decoder with deconvolution layers, as described in Figure B.2. Our input time series are of length 384 and we

[1] we only kept clients for which survey data was available

choose a bottleneck of $d = 20 << 384$. The CAE is trained end-to-end by minimizing the loss

$$\min \sum_{i=1}^{N} \mathcal{L}(x^i, \hat{x}^i) \tag{B.1}$$

where $\hat{X}_i$ is the output of the CAE and L is the ℓ^2-loss. Our CAE is then used as a non-linear dimensionality reduction. We perform the clustering *a posteriori* using K-medoids algorithm.

We work with limited data so the architecture is kept small, with 2 convolutional layers, one dense layer and two deconvolutional layers. The used architecture for the CAE is the following for both experiments:

- Conv1D: 64 filters of length 3, stride 2, activation *elu*
- BatchNormalization
- Conv1D: 128 filters of length 5, stride 2, activation *elu*
- BatchNormalization
- Flatten
- FullyConnected: 100 units, activation *elu*
- FullyConnected: 20 units, activation *linear*
- FullyConnected: 128×96 units, activation *elu*
- BatchNormalization
- DeConv1D: 128 filters of length 5, activation *elu*
- BatchNormalization
- DeConv1D: 64 filters of length 3, activation *elu*
- Conv1D: 1 filters of length 3, activation *tanh*

We include BatchNormalization layers for faster learning. Each implementation has been made using Python with Keras and Tensorflow for the convolutional autoencoder. In order to train a convolutional network, it is known that one must normalize the data. For time series, we cannot normalize each feature, instead the z-normalization is very commonly used, *ie* normalizing each time series by subtracting the mean and dividing by its standard deviation. We will see that it is not always the most efficient way to proceed in Section 3.4.

B.2.2 Compared methods

We compare our approach to other feature extraction methods. We compare several feature extraction approaches as a pre-processing for the K-medoids algorithm or different time series clustering algorithm:

- **K-medoids:** K-medoids performed on raw data (divided by the mean)
- **PCA+K-medoids:** only the 20 first components of PCA are used
- **Haar+IK-means:** the coefficients of a Discrete Wavelet Transform with Haar wavelet are used in an interactive K-Means as defined in [180]
- **Dictionary Learning + K-Medoids**: K-medoids performed on the coefficients learnt from a dictionary learnt with non-negative matrix factorization
- **CAE+K-medoids:** K-medoids performed on latent vector from the convolutional autoencoder

B.2.3 Outliers

One of the main issues of traditional clustering techniques is the robustness to outliers. But defining an outlier in a dataset is a challenging task. Different approaches have been proposed: Mahalanobis distance, Outlier Detection using Indegree Number, ... Here, we use the Local Outlier Factor metrics[34].

To define this metric we have to define a distance on our initial space. Here we choose the euclidean distance. A *k-NN* graph is computed on the dataset. Defining the *local reachibility density* of an example x_i as :

$$lrd(x_i) = \left(\frac{\sum_{x_j \in \mathcal{N}(x_i)} \max(d_k(x_j), d(x_i, x_j))}{k} \right)^{-1} \tag{B.2}$$

where $d_k(x_j) = \max_{x_k \in \mathcal{N}(x_i)} d(x_j, x_k)$ and $\mathcal{N}(x_i)$ is the neighbourhood of x_i

The **Local Outlier Factor** is defined as:

$$LOF(x_i) = \frac{\sum_{x_j \in \mathcal{N}(x_i)} lrd(x_j)}{klrd(x_i)} \tag{B.3}$$

The main idea behind this metric is that it allows to define outliers locally. Therefore points that would be defined as outliers with other methods such as Mahalanobis because they belong to a small or sparse cluster will be considered as normal with this metric. A threshold is manually defined such that every point with a LOF higher than the threshold is considered as an outlier. We use this definition to determine the number of outliers per cluster which quantifies the *outlyingness* of a cluster.

On the CER dataset, using this LOF on raw data shows that outliers often are secondary houses with extremely high consumption. Ideally, we would want a clustering method that groups those outliers together. We will see that most feature transformations fail to do so but features extracted from a CAE give surprisingly good results.

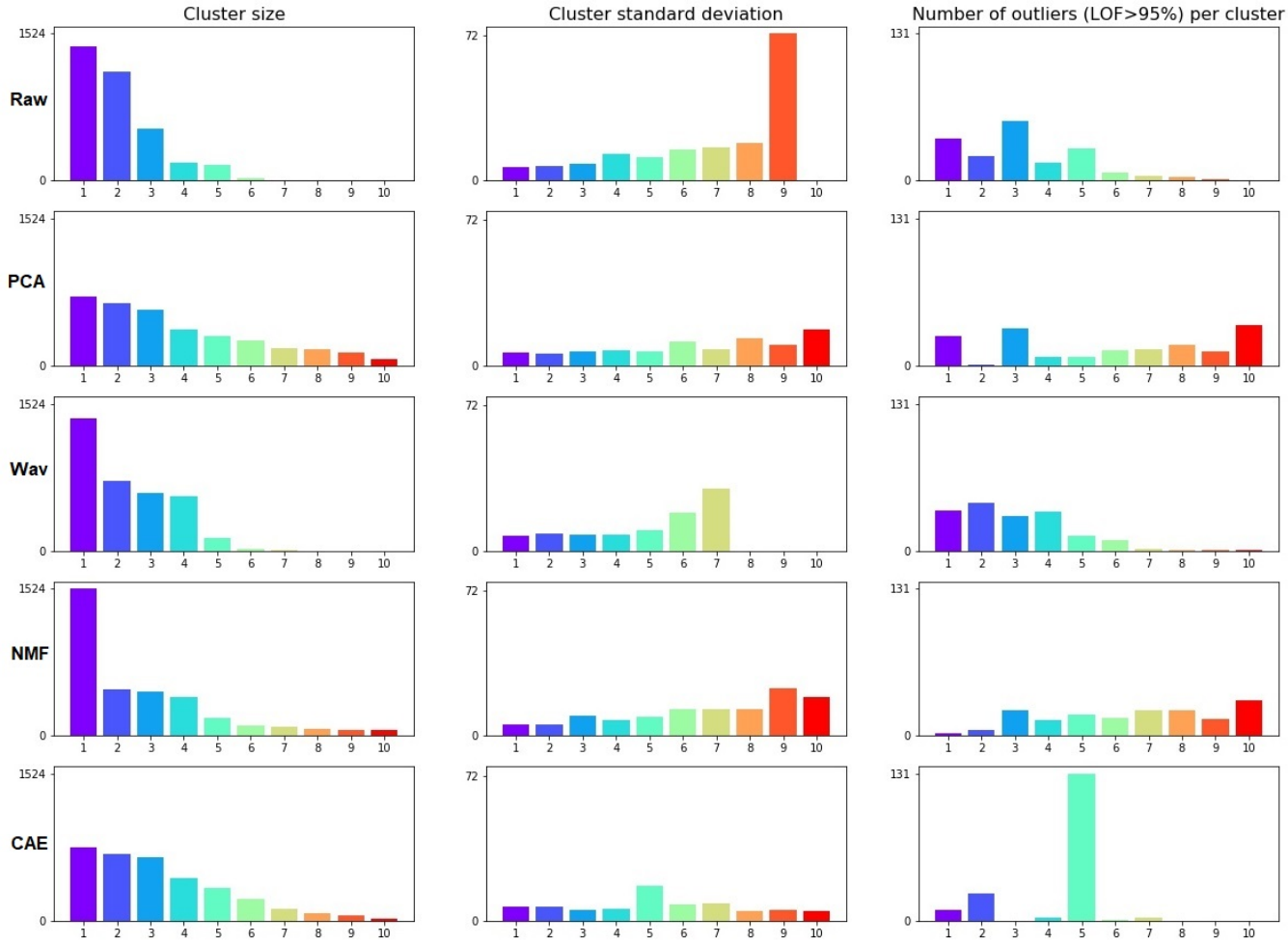


Figure B.3: Left: number of clients per cluster ; Center: number of elements per clusters ; Right: number of outliers in each cluster (an element is an outlier if its Local Outlier Factor is above the 95 % quantile)

B.3 Results

Using the elbow method we find that a good number of clusters would be 10 or 16. On Figure B.3 are plotted the size of each cluster and the number of outliers per cluster. An outlier is defined by its Local Outlier Factor and here we set the number of outliers at 5% of the dataset.

One can see that most methods do not cope well with outliers: either the outliers are spread between clusters or they are all gathered in the major class. But the proposed method allows to isolate a class with high dispersion which gathers the outliers. Then the rest of the data can be clustered into finer clusters.

Clusters have an interpretation and the centroids are shown in B.4. Two clusters of SMEs come out and residential clients are clustered depending on other features: the peak in consumption during the 2010 winter or Christmas for example, which is higher for clients with electrical heater.

The outlier cluster is characterized by many clients with abnormal consumption patterns (often very small or

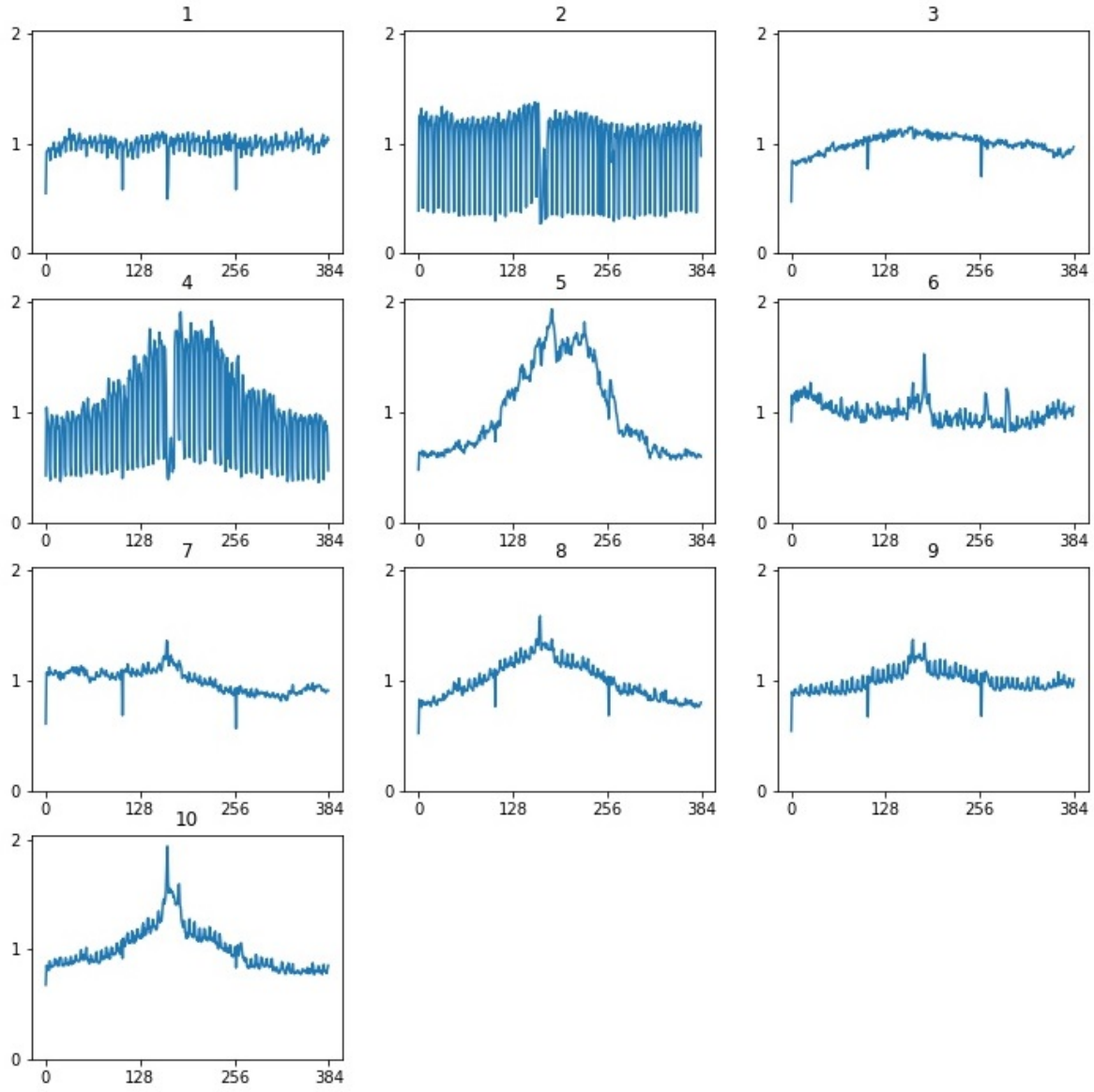


Figure B.4: Centroids of each cluster found with the CAE+K-Medoids method

with few very high values). Tests were done with other numbers of clusters and the isolation of one *outlier class* remains. After 20 clusters the class is split into two. Experiments have also been conducted with K-Means and Self-Organizing Maps (SOM). Similar results were obtained with the CAE whereas performance with other feature extraction dropped. This tends to show that convolutional autoencoder is a feature extractor for time series with very nice properties.

We expected the results to be similar with Non-Negative Matrix Factorization but while it separates very well the residential households from SMEs (see Figure B.5), using the coefficients for clustering gives clusters polluted by outliers. The interpretability is better than the CAE, where it is hard to know what neurons correspond to what. We could only identify one variable corresponding to the week-end consumption drop for SMEs while NMF extracts

interpretable components (week-end drop, higher consumption in January given the extreme cold in Ireland, ...).

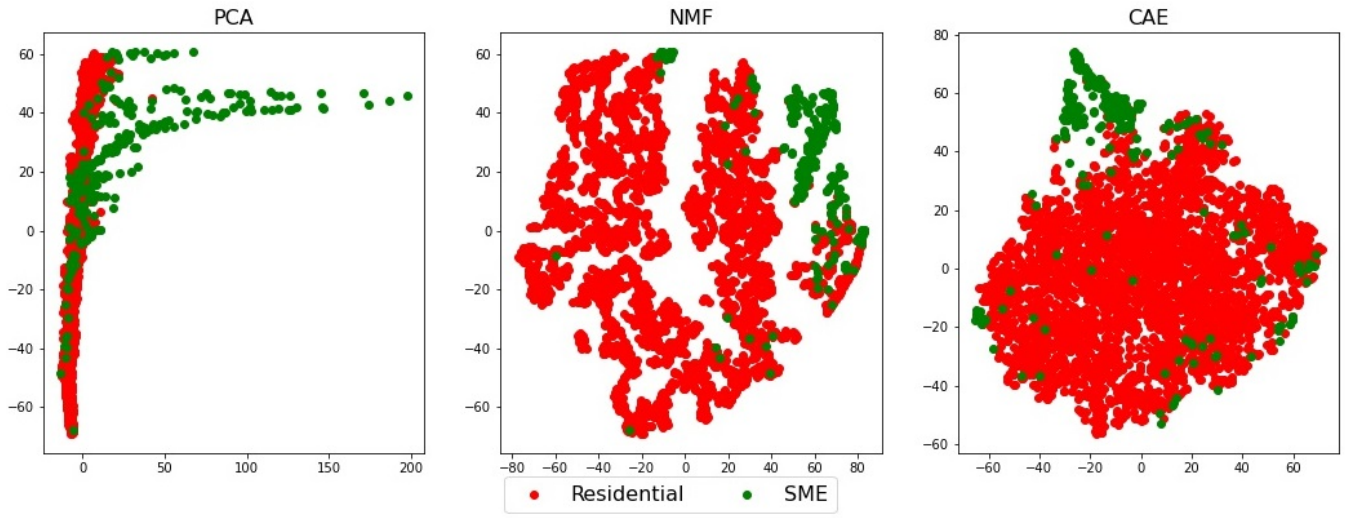


Figure B.5: SME vs Residentials using (a) PCA representation (b) NMF representation (c) TSNE representation of the latent variables obtained by CAE

Appendix C

Implementations

During this thesis, we developed a number of methods and algorithms. Some of those methods are publicly available, integrated into the Python *adapt* package or in the public git repositories attached to the papers. Every implementation made in this thesis was done in Python. Moreover, apart from the experiments on electricdata of Chapter 4 experiments were conducted on a machine with two Quadro P6000 GPUs with 24Gb RAM. For the experiments on electricdata of Chapter 4, we used the High-performance computing server of EDF, with several GPUs.

C.1 Public implementations

Adapt package During this thesis, we contributed to the *adapt* [53] package. This package collects traditional methods for domain adaptation, such as **KMM**, **DANN**, **TrAdaBoost**, ... The whole documentation can be found at https://antoinedemathelin.github.io/adapt/_build/html/index.html. We also provided some dataset loading utilities, for common domain adaptation datasets. Those datasets are:

- **Amazon Multi-Domain Sentiment Dataset:** [1] this dataset, firstly used in [29], is made of textual Amazon reviews from 25 categories of objects, where the variable to predict is the rating of the review. We propose raw data, data transformed with TF-IDF and with Word2Vec (Google300 and Glove25).
- **Digits Dataset:** this dataset merges several datasets of digit recognition: MNIST, MNIST-M, SVHN, Synth and USPS.
- **Office-Caltech-Amazon dataset:**[2] this dataset is an object recognition dataset, where images of objects are captured in different environments and with different means. Raw images and extracted features are available.

[1] https://www.cs.jhu.edu/~mdredze/datasets/sentiment/
[2] https://people.eecs.berkeley.edu/~jhoffman/domainadapt/

- **Human Activity Recognition dataset:** [9] 30 subjects were monitored using a smartphone while performing 6 daily activities. Raw time series and extracted features are available.

Those datasets are the most commonly used datasets in the domain adaptation community. It should still be noted that for the Amazon Multi-Domain Sentiment Dataset, most papers use the same pre-processing step with TF-IDF. We noted in our experiments that better results can be obtained using a pre-trained Word2Vec, without any adaptation. Similarly, for image datasets, state-of-the art results were obtained by using adversarial adaptation methods based on neural networks. Those neural networks were initialized using a pre-trained network like VGG16. This is already a form of transfer learning and it makes it hard to evaluate the contribution of the pre-training and the subsequent adaptation.

Hypothesis-Discrepancy The method developed in Chapter 4 is publicly available in the following github repository: https://github.com/GRichard513/ADisc-MSDA. We also give the code for the experiments on synthetic data and public datasets for reproducibility. The implementation of our method is based on PyTorch[3]. We give the implementation for those public datasets but the extension to other datasets is straightforward: the user simply needs to change the parameters of the class *Disc_MSDANet*: in particular, the *feature_extractor* parameter allows the user to define its own feature extractor network, which will be application-specific. The feature extractor needs to be implemented using pytorch.

Covariance-based Transfer Learning For this implementation, we based our implementation on the PyManopt [173] package. We simply defined our own functions to compute the Log-Euclidean Kernel matrix and the gradient steps efficiently by caching the matrix logarithm of the input covariance matrices in memory. We will make the implementations public after publication of the method.

C.2 Other implementations

C.2.1 List of statistical features extracted (Chapter 3)

We used the library *tsfresh* (https://tsfresh.readthedocs.io/en/latest/index.html) to extract features using their *EfficientParameters* class. The full extracted features are available in Table C.1. Most features have understandable names and full details are available in the *tsfresh* documentation.

[3] https://pytorch.org/

abs_energy	absolute_sum_of_changes
agg_autocorrelation_mean	agg_autocorrelation_median
agg_autocorrelation_var	ar_coefficient_k_10_coeff_0
ar_coefficient_k_10_coeff_1	ar_coefficient_k_10_coeff_2
ar_coefficient_k_10_coeff_3	ar_coefficient_k_10_coeff_4
autocorrelation_lag_0	autocorrelation_lag_1
autocorrelation_lag_2	autocorrelation_lag_3
autocorrelation_lag_4	autocorrelation_lag_5
autocorrelation_lag_6	autocorrelation_lag_7
autocorrelation_lag_8	autocorrelation_lag_9
binned_entropy_max_bins_10	c3_lag_1
c3_lag_2	c3_lag_3
cid_ce_normalize_False	cid_ce_normalize_True
count_above_mean	count_below_mean
energy_ratio_by_chunks_num_segments_10_segment_focus_0	energy_ratio_by_chunks_num_segments_10_segment_focus_1
energy_ratio_by_chunks_num_segments_10_segment_focus_2	energy_ratio_by_chunks_num_segments_10_segment_focus_3
energy_ratio_by_chunks_num_segments_10_segment_focus_4	energy_ratio_by_chunks_num_segments_10_segment_focus_5
energy_ratio_by_chunks_num_segments_10_segment_focus_6	energy_ratio_by_chunks_num_segments_10_segment_focus_7
energy_ratio_by_chunks_num_segments_10_segment_focus_8	energy_ratio_by_chunks_num_segments_10_segment_focus_9
first_location_of_maximum	first_location_of_minimum
has_duplicate	index_mass_quantile_q_0.1
index_mass_quantile_q_0.2	index_mass_quantile_q_0.3
index_mass_quantile_q_0.4	index_mass_quantile_q_0.6
index_mass_quantile_q_0.7	index_mass_quantile_q_0.8
index_mass_quantile_q_0.9	kurtosis
last_location_of_maximum	last_location_of_minimum
length ”	linear_trend_attr_””intercept””” ”
linear_trend_attr_””pvalue””” ”	linear_trend_attr_””rvalue””” ”
linear_trend_attr_””slope””” ”	linear_trend_attr_””stderr”””
longest_strike_above_mean	longest_strike_below_mean
maximum	mean
mean_ab_change	mean_change
mean_second_derivative_central	median
minimum	number_crossing_m_m_-1
number_crossing_m_m_0	number_crossing_m_m_1
number_peaks_n_1	number_peaks_n_10
number_peaks_n_3	number_peaks_n_5
number_peaks_n_50	partial_autocorrelation_lag_0
partial_autocorrelation_lag_1	partial_autocorrelation_lag_2
partial_autocorrelation_lag_3	partial_autocorrelation_lag_4
partial_autocorrelation_lag_5	partial_autocorrelation_lag_6
partial_autocorrelation_lag_7	partial_autocorrelation_lag_8
partial_autocorrelation_lag_9	quantile_q_0.1
quantile_q_0.2	quantile_q_0.3
quantile_q_0.4	quantile_q_0.6
quantile_q_0.7	quantile_q_0.8
quantile_q_0.9	range_count_max_1_min_-1
ratio_value_number_to_time_series_length	skewness
spkt_welch_density_coeff_2	spkt_welch_density_coeff_5
spkt_welch_density_coeff_8	standard_deviation
sum_of_reoccurring_data_points	sum_of_reoccurring_values
sum_values	time_reversal_asymmetry_statistic_lag_1
time_reversal_asymmetry_statistic_lag_2	time_reversal_asymmetry_statistic_lag_3
variance	variance_larger_than_standard_deviation

Table C.1: List of eaxtracted features

C.3 Additional Experiments of Chapter 3

In order to compare with existing methods, we report the results available online [**?**] using Nearest Neighbour classifiers associated with Euclidean Distance (ED-NN), Dynamic Time Warping (DTW-NN) and Dynamic Time Warping with a learned window (WDTW-NN). The full results are available in Table C.2 and we plot the rank distribution of each algorithm on Figure C.1.

The first observation is that different data preparations affect the performance of a convolutional neural network. Moreover there is no universal choice of normalization to get the best classification accuracy. For most datasets, the Ens-Norm Network achieves better accuracy than the simple DenseNets, which indicates that this architecture is able to derive the best from both standardizations. Standardization seems to be more efficient than min-max normalization in general.

Overall we achieve slightly better results than the existing nearest neighbour classifiers. We do believe that better accuracies can be achieved, notably with ensemble-based methods, whose voting scheme could even include neural networks. We conducted experiments with ResNets and FCN architectures that generally did not lead to better accuracies but showed similar observations for the impact of pre-processing.

Dataset	ED-NN	WDTW-NN	DTW-NN	GS	GN	IS	IN	BC-GS	BC-IS	IS-Feat	IS-GS-NEN
AllGestureWiimoteX	51,57%	71,71%	71,57%	**77,89%**	68,34%	73,34%	66,89%	76,83%	67,14%	68,34%	77,83%
AllGestureWiimoteY	56,86%	**73,00%**	72,86%	72,60%	72,91%	73,34%	72,26%	75,80%	68,86%	72,91%	72,91%
AllGestureWiimoteZ	45,43%	65,14%	64,29%	70,06%	67,49%	69,40%	66,20%	75,80%	50,86%	67,49%	**76,31%**
BME	83,33%	98,00%	90,00%	92,13%	92,13%	95,33%	96,27%	94,27%	**99,07%**	94,53%	96,27%
Chinatown	95,36%	95,36%	95,65%	97,97%	97,97%	97,97%	97,97%	97,68%	97,97%	97,97%	**98,26%**
Crop	71,17%	71,17%	66,52%	78,30%	74,52%	76,01%	74,90%	78,72%	75,19%	74,52%	**79,26%**
DodgerLoopDay	55,00%	58,75%	50,00%	55,00%	40,75%	61,25%	42,75%	49,25%	40,75%	55,75%	**62,75%**
DodgerLoopGame	88,41%	**92,75%**	87,68%	86,96%	88,41%	84,35%	85,65%	88,41%	83,33%	88,41%	88,26%
DodgerLoopWeekend	**98,55%**	97,83%	94,93%	88,12%	95,36%	92,75%	92,75%	89,71%	93,77%	95,36%	92,75%
EOGHorizontalSignal	41,71%	47,51%	50,28%	61,27%	58,62%	58,62%	59,12%	59,12%	52,98%	61,49%	**61,82%**
EOGVerticalSignal	44,20%	47,51%	44,75%	48,67%	46,35%	46,24%	44,20%	46,35%	42,38%	**49,12%**	48,84%
Fungi	82,26%	82,26%	**83,87%**	53,87%	68,06%	64,84%	72,37%	28,82%	49,78%	68,06%	64,52%
GestureMidAirD1	57,69%	**63,85%**	56,92%	59,54%	55,85%	62,15%	58,15%	58,15%	59,23%	59,23%	63,08%
GestureMidAirD2	49,23%	60,00%	**60,77%**	46,46%	57,23%	59,23%	54,46%	42,31%	42,92%	59,23%	59,85%
GestureMidAirD3	34,62%	**37,69%**	32,31%	19,85%	30,31%	33,54%	30,77%	21,38%	26,46%	34,46%	34,31%
GesturePebbleZ1	73,26%	82,56%	79,07%	59,88%	63,37%	84,19%	62,21%	62,33%	76,74%	**84,77%**	84,42%
GesturePebbleZ2	67,09%	**77,85%**	67,09%	59,62%	55,06%	73,80%	61,01%	67,97%	73,42%	71,27%	74,43%
GunPointAgeSpan	89,87%	96,52%	91,77%	98,67%	99,37%	99,56%	**99,62%**	98,61%	98,10%	99,37%	**99,62%**
GunPointMaleVersusFemale	97,47%	97,47%	99,68%	**99,81%**	99,37%	99,05%	99,37%	99,68%	96,96%	99,37%	**99,81%**
GunPointOldVersusYoung	95,24%	96,51%	83,81%	**100,00%**	96,70%	97,08%	96,38%	**100,00%**	95,11%	96,70%	99,87%
HouseTwenty	66,39%	94,12%	92,44%	94,12%	92,44%	42,02%	42,02%	89,92%	42,02%	42,02%	**94,79%**
InsectEPGRegularTrain	67,87%	82,73%	87,15%	**100,00%**	99,68%	97,75%	98,96%	**100,00%**	96,79%	99,68%	100,00%
InsectEPGSmallTrain	66,27%	69,48%	73,49%	35,74%	35,74%	95,98%	47,39%	35,74%	90,68%	**96,71%**	96,47%
MelbournePedestrian	84,82%	84,82%	79,06%	96,44%	90,25%	90,53%	90,43%	97,02%	90,36%	95,31%	**97,11%**
PLAID	53,63%	83,61%	83,80%	83,99%	83,09%	84,21%	84,25%	83,09%	69,42%	83,09%	**84,25%**
PickupGestureWiimoteZ	56,00%	66,00%	66,00%	**76,00%**	66,80%	70,40%	60,80%	72,40%	47,60%	70,00%	75,60%
PigAirwayPressure	5,77%	9,62%	10,58%	**18,17%**	6,15%	10,48%	6,63%	15,10%	10,87%	13,56%	17,40%
PigArtPressure	12,50%	19,71%	24,52%	51,06%	16,73%	47,50%	19,52%	49,90%	44,33%	16,73%	**52,02%**
PigCVP	8,17%	15,87%	15,38%	47,21%	18,65%	50,58%	19,33%	46,44%	**52,69%**	51,15%	52,31%
PowerCons	93,33%	92,22%	87,78%	95,44%	89,89%	90,56%	89,22%	94,78%	89,00%	89,89%	**96,89%**
Rock	**84,00%**	**84,00%**	60,00%	71,60%	62,00%	59,60%	62,00%	73,20%	62,40%	62,00%	62,00%
SemgHandGenderCh2	76,17%	**84,50%**	80,17%	79,67%	65,83%	83,40%	35,00%	82,93%	81,77%	82,77%	84,00%
SemgHandMovementCh2	36,89%	**63,78%**	58,44%	54,53%	40,62%	44,71%	39,24%	49,60%	46,31%	40,62%	39,24%
SemgHandSubjectCh2	40,44%	**80,00%**	72,67%	71,69%	70,98%	74,71%	73,24%	70,27%	69,29%	72,22%	73,24%
ShakeGestureWiimoteZ	60,00%	84,00%	86,00%	88,40%	88,40%	84,40%	86,40%	87,20%	72,40%	88,40%	**89,20%**
SmoothSubspace	90,67%	94,67%	82,67%	98,00%	97,33%	96,53%	97,33%	**98,67%**	96,00%	97,33%	97,33%
UMD	76,39%	97,22%	**99,31%**	**99,31%**	98,61%	98,61%	98,61%	**99,31%**	**99,31%**	98,61%	**99,31%**

Table C.2: Accuracies for each non-normalized UCR dataset

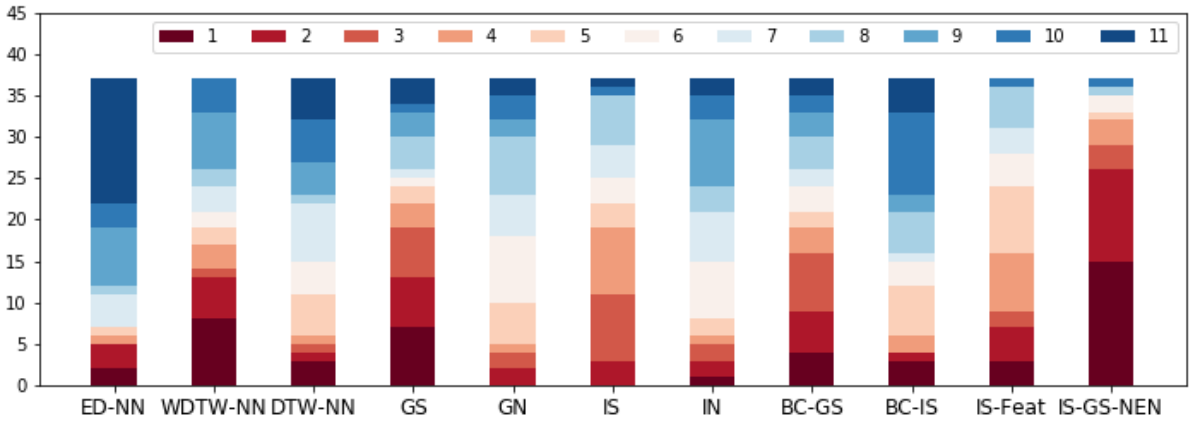


Figure C.1: Rank distribution of each method over UCR non-normalized datasets. Each bar corresponds to a method. Red shows a high rank and blue a low rank.

C.4 Details about implementations of Chapter 4

Pre-processing for NILM applications

As raw consumption data has values from 0W to 20000W, it needs to be normalized before being fed to a neural network. The best normalization was a a log-transform on the input data, and then a global normalization. The log-transform makes the training less sensitive to outliers and the global normalization is a simple rescaling that allows neural networks to train. During test time, we use the statistics from training data to normalize test input data.

$$
\begin{aligned}
X_{log} &= \log X_{train} + \epsilon \\
X'_{train} &= \frac{X_{log} - mean(X_{log})}{std(X_{log})}
\end{aligned}
\tag{C.1}
$$

where ϵ is a small precision value. Similarly, the output y was normalized as it allowed for better training behaviour from the network.

$$
\begin{aligned}
y_{train} &= \frac{y_{train} - mean(y_{train})}{std(y_{train})} \\
y_{test} &= \frac{y_{test} - mean(y_{train})}{std(y_{train})}
\end{aligned}
\tag{C.2}
$$

To estimate the errors made, we denormalize the prediction and the true value of consumptions.

Architectures for other applications

Amazon The architecture is kept the same for each method:

- **Feature extractor:**
  - $Linear(5000, 500, LeakyRELU)$,
  - $Dropout(0.1)$
  - $Linear(500, 20, LeakyRELU)$,
  - $Dropout(0.1)$
- **Predictor/Discriminator:**
  - $Linear(20, 1)$

Digits The architecture is kept the same for each method:

- **Feature extractor:**
  - $Conv2d(1, 64, 3, padding = 1)$ with $LeakyReLU$
  - $MaxPool2d(2)$
  - $Conv2d(64, 128, 3, padding = 1)$ with $LeakyReLU$
  - $MaxPool2d(2)$
  - $Conv2d(128, 256, 3, padding = 1)$ with $LeakyReLU$

  - $MaxPool2d(2)$
  - $Flatten()$

- **Predictor / Discriminator:**
  - $Linear(2304, 100)$ with $LeakyReLU$
  - $Dropout(0.2)$
  - $Linear(100, 10)$ (predictor) / $Linear(100, 2)$ (discriminator)
  - $Softmax(dim = -1)$

Bibliography

[1] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[2] B. Adlam, C. Cortes, M. Mohri, and N. Zhang. Learning gans and ensembles using discrepancy. In *Advances in Neural Information Processing Systems*, pages 5788–5799, 2019.

[3] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

[4] H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, and M. Marchand. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*, 2014.

[5] A. H. Al-Mohy and N. J. Higham. Computing the fréchet derivative of the matrix exponential, with an application to condition number estimation. *SIAM Journal on Matrix Analysis and Applications*, 30(4):1639–1657, 2009.

[6] R. Aljundi, R. Emonet, D. Muselet, and M. Sebban. Landmarks-based kernelized subspace alignment for unsupervised domain adaptation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 56–63. IEEE Computer Society, jun 2015.

[7] E. Amouroux, T. Huraux, F. Sempé, N. Sabouret, and Y. Haradji. Simulating human activities to investigate household energy consumption. 2013.

[8] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853, 2005.

[9] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *Esann*, volume 3, page 3, 2013.

[10] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. *Advances in neural information processing systems*, 19:41–48, 2006.

[11] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Log-euclidean metrics for fast and simple calculus on diffusion tensors. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 56(2):411–421, 2006.

[12] F. Bach, R. Jenatton, J. Mairal, G. Obozinski, et al. Structured sparsity through convex optimization. *Statistical Science*, 27(4):450–468, 2012.

[13] A. Bagnall and G. Janacek. A run length transformation for discriminating between auto regressive time series. *Journal of classification*, 31(2):154–178, 2014.

[14] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[15] M. Baktashmotlagh, M. Harandi, and M. Salzmann. Distribution-matching embedding for visual domain adaptation. *The Journal of Machine Learning Research*, 17(1):3760–3789, 2016.

[16] A. Barachant, S. Bonnet, M. Congedo, and C. Jutten. Multiclass brain–computer interface classification by riemannian geometry. *IEEE Transactions on Biomedical Engineering*, 59(4):920–928, 2011.

[17] A. Barachant, S. Bonnet, M. Congedo, and C. Jutten. Classification of covariance matrices using a riemannian-based kernel for bci applications. *Neurocomputing*, 112:172–178, 2013.

[18] F. Barbaresco. Interactions between symmetric cone and information geometries: Bruhat-tits and siegel spaces models for high resolution autoregressive doppler imagery. In *LIX Fall Colloquium on Emerging Trends in Visual Computing*, pages 124–163. Springer, 2008.

[19] O. Barndorff-Nielsen and G. Schou. On the parametrization of autoregressive models by partial autocorrelations. *Journal of multivariate Analysis*, 3(4):408–419, 1973.

[20] K. Bascol, R. Emonet, and E. Fromont. Improving domain adaptation by source selection. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 3043–3047. IEEE, 2019.

[21] N. Batra, H. Dutta, and A. Singh. Indic: Improved non-intrusive load monitoring using load division and calibration. In *2013 12th International Conference on Machine Learning and Applications*, volume 1, pages 79–84. IEEE, 2013.

[22] J. Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000.

[23] M. G. Baydogan, G. Runger, and E. Tuv. A bag-of-features framework to classify time series. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2796–2802, 2013.

[24] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137–144, 2006.

[25] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[26] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, USA:, 1994.

[27] R. Bhatia. *Positive definite matrices*, volume 24. Princeton university press, 2009.

[28] S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th international conference on Machine learning*, pages 81–88, 2007.

[29] 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*, pages 440–447, 2007.

[30] J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman. Learning bounds for domain adaptation. In *Advances in neural information processing systems*, pages 129–136, 2008.

[31] P. Bloomfield. *Fourier analysis of time series: an introduction*. John Wiley & Sons, 2004.

[32] N. Boumal and P.-A. Absil. A discrete regression method on manifolds and its application to data on so (n). *IFAC Proceedings Volumes*, 44(1):2284–2289, 2011.

[33] O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of machine learning research*, pages 499–526, March 2002.

[34] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

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

[36] Y. Cabanes, F. Barbaresco, M. Arnaudon, and J. Bigot. Toeplitz hermitian positive definite matrix machine learning based on fisher metric. In *International Conference on Geometric Science of Information*, pages 261–270. Springer, 2019.

[37] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[38] J. Cavazza, P. Morerio, and V. Murino. When kernel methods meet feature learning: Log-covariance network for action recognition from skeletal data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 33–40, 2017.

[39] R. Chattopadhyay, Q. Sun, W. Fan, I. Davidson, S. Panchanathan, and J. Ye. Multisource domain adaptation and its application to early detection of fatigue. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(4):1–26, 2012.

[40] M. Chen, Z. Xu, K. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, 1, 06 2012.

[41] S. Chen, S. Ma, A. Man-Cho So, and T. Zhang. Proximal gradient method for nonsmooth optimization over the stiefel manifold. *SIAM Journal on Optimization*, 30(1):210–239, 2020.

[42] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[43] C. Cortes, M. Mohri, and A. Rostamizadeh. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13(1):795–828, 2012.

[44] C. Cortes, M. Mohri, and A. Muñoz Medina. Adaptation algorithm and theory based on generalized discrepancy. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 169–178, 2015.

[45] C. Cortes, X. Gonzalvo, V. Kuznetsov, M. Mohri, and S. Yang. Adanet: Adaptive structural learning of artificial neural networks. In *International conference on machine learning*, pages 874–883. PMLR, 2017.

[46] C. Cortes, M. Mohri, and A. M. Medina. Adaptation based on generalized discrepancy. *The Journal of Machine Learning Research*, 20(1):1–30, 2019.

[47] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1853–1865, 2017.

[48] K. Crammer, M. Kearns, and J. Wortman. Learning from multiple sources. *Journal of Machine Learning Research*, 9(Aug):1757–1774, 2008.

[49] Z. Cui, W. Chen, and Y. Chen. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995*, 2016.

[50] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200, 2007.

[51] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

[52] H. Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009.

[53] A. de Mathelin. Adapt awesome domain adaptation package toolbox, 2020. URL https://antoinedemathelin.github.io/adapt/_build/html/index.html.

[54] A. de Mathelin, G. Richard, M. Mougeot, and N. Vayatis. Adversarial weighting for domain adaptation in regression. *arXiv preprint arXiv:2006.08251*, 2020.

[55] A. Delfosse, G. Hébrail, and A. Zerroug. Deep learning applied to nilm: is data augmentation worth for energy disaggregation?

[56] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[57] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[58] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.

[59] B. Dumitrescu and P. Irofti. *Dictionary learning algorithms and applications*. Springer, 2018.

[60] L. Duong, T. Cohn, S. Bird, and P. Cook. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 845–850, 2015.

[61] M. D'Incecco, S. Squartini, and M. Zhong. Transfer learning for non-intrusive load monitoring. *IEEE Transactions on Smart Grid*, 11(2):1419–1429, 2019.

[62] H. A. El-Ghaish, A. A. Shoukry, and M. E. Hussein. Covp3dj: Skeleton-parts-based-covariance descriptor for human action recognition. In *VISIGRAPP (5: VISAPP)*, pages 343–350, 2018.

[63] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117, 2004.

[64] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. Transfer learning for time series classification. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1367–1376. IEEE, 2018.

[65] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

[66] H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, 2020.

[67] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *2013 IEEE International Conference on Computer Vision*, pages 2960–2967, 2013.

[68] B. Fernando, T. Tommasi, and T. Tuytelaars. Joint cross-domain classification and subspace learning for unsupervised adaptation. *Pattern Recogn. Lett.*, 65(C):60–66, Nov. 2015. ISSN 0167-8655.

[69] C. for Energy Regulation (CER). Cer smart metering project - electricity customer behaviour trial [dataset], 2009-2010.

[70] J. H. Friedman. Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67, 1991.

[71] C. Gallicchio and A. Micheli. Deep echo state network (deepesn): A brief survey. *arXiv preprint arXiv:1712.04323*, 2017.

[72] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17:59:1–59:35, 2016.

[73] P. Germain, A. Habrard, F. Laviolette, and E. Morvant. A new pac-bayesian perspective on domain adaptation. In *International conference on machine learning*, pages 859–868, 2016.

[74] P. Germain, A. Habrard, F. Laviolette, and E. Morvant. Pac-bayes and domain adaptation. *Neurocomputing*, 379:379–397, 2020.

[75] X. Glorot, A. Bordes, and Y. Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 513–520, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

[76] B. Gong, K. Grauman, and F. Sha. *Geodesic Flow Kernel and Landmarks: Kernel Methods for Unsupervised Domain Adaptation*, pages 59–79. Springer International Publishing, Cham, 2017.

[77] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[78] A. Gretton. Covariate shift by kernel mean matching. In *NIPS 2009 Workshop on Transfer Learning for Structured Data (TLSD-09)*, 2009.

[79] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[80] I. Guyon and A. Elisseeff. An introduction of variable and feature selection. *J. Machine Learning Research Special Issue on Variable and Feature Selection*, 3:1157 – 1182, 01 2003.

[81] D. Hallac, S. Vare, S. Boyd, and J. Leskovec. Toeplitz inverse covariance-based clustering of multivariate time series data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 215–223, 2017.

[82] L. Han and Y. Zhang. Learning tree structure in multi-task learning. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 397–406, 2015.

[83] M. Harandi, M. Salzmann, and R. Hartley. Dimensionality reduction on spd manifolds: The emergence of geometry-aware methods. *IEEE transactions on pattern analysis and machine intelligence*, 40(1):48–62, 2017.

[84] M. T. Harandi, C. Sanderson, R. Hartley, and B. C. Lovell. Sparse coding and dictionary learning for symmetric positive definite matrices: A kernel approach. In *European Conference on Computer Vision*, pages 216–229. Springer, 2012.

[85] G. W. Hart. Residential energy monitoring and computerized surveillance via utility power flows. *IEEE Technology and Society Magazine*, 8(2):12–16, 1989.

[86] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[87] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[88] J. Hoffman, M. Mohri, and N. Zhang. Algorithms and theory for multiple-source adaptation. In *Advances in Neural Information Processing Systems*, pages 8246–8256, 2018.

[89] J.-F. Hu, W.-S. Zheng, J. Lai, and J. Zhang. Jointly learning heterogeneous features for rgb-d activity recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5344–5352, 2015.

[90] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[91] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. J. Smola. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608, 2007.

[92] S.-W. Huang, C.-T. Lin, S.-P. Chen, Y.-Y. Wu, P.-H. Hsu, and S.-H. Lai. Auggan: Cross domain adaptation with gan-based data augmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 718–731, 2018.

[93] Z. Huang, R. Wang, X. Li, W. Liu, S. Shan, L. Van Gool, and X. Chen. Geometry-aware similarity learning on spd manifolds for visual recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 28 (10):2513–2523, 2017.

[94] R. Ievgen and B. Younés. Random subspaces nmf for unsupervised transfer learning. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 3901–3908. IEEE, 2014.

[95] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[96] L. Jacob, J.-p. Vert, and F. R. Bach. Clustered multi-task learning: A convex formulation. In *Advances in neural information processing systems*, pages 745–752, 2009.

[97] A. Jalali, S. Sanghavi, C. Ruan, and P. K. Ravikumar. A dirty model for multi-task learning. In *Advances in neural information processing systems*, pages 964–972, 2010.

[98] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, and M. Harandi. Kernel methods on riemannian manifolds with gaussian rbf kernels. *IEEE transactions on pattern analysis and machine intelligence*, 37(12):2464–2477, 2015.

[99] B. Jeuris and R. Vandebril. The kahler mean of block-toeplitz matrices with toeplitz structured blocks. *SIAM Journal on Matrix Analysis and Applications*, 37(3):1151–1175, 2016.

[100] J. Kelly and W. Knottenbelt. Neural nilm: Deep neural networks applied to energy disaggregation. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, pages 55–64, 2015.

[101] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *VLDB*, volume 4, pages 180–191. Toronto, Canada, 2004.

[102] J. Kolter, S. Batra, and A. Ng. Energy disaggregation via discriminative sparse coding. *Advances in neural information processing systems*, 23:1153–1161, 2010.

[103] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[104] S. Kuroki, N. Charoenphakdee, H. Bao, J. Honda, I. Sato, and M. Sugiyama. Unsupervised domain adaptation based on source-guided discrepancy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4122–4129, 2019.

[105] M. Längkvist, L. Karlsson, and A. Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

[106] A. Le Guennec, S. Malinowski, and R. Tavenard. Data augmentation for time series classification using convolutional neural networks. 2016.

[107] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[108] O. Ledoit and M. Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis*, 88(2):365–411, 2004.

[109] D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13:556–562, 2000.

[110] X. Li, S. Chen, Z. Deng, Q. Qu, Z. Zhu, and A. M. C. So. Nonsmooth optimization over stiefel manifold: Riemannian subgradient methods. *arXiv preprint arXiv:1911.05047*, 2019.

[111] Y. Lin, Y. Lee, and G. Wahba. Support vector machines for classification in nonstandard situations. *Machine learning*, 46(1-3):191–202, 2002.

[112] J. Lines, S. Taylor, and A. Bagnall. Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 1041–1046. IEEE, 2016.

[113] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu. Transfer feature learning with joint distribution adaptation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[114] M. Long, J. Wang, G. Ding, S. Pan, and P. Yu. Adaptation regularization: A general framework for transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 26:1076–1089, 05 2014.

[115] K. Lounici, M. Pontil, A. B. Tsybakov, and S. Van De Geer. Taking advantage of sparsity in multi-task learning. *arXiv preprint arXiv:0903.1468*, 2009.

[116] M. Lovrić, M. Milanović, and M. Stamenković. Algoritmic methods for segmentation of time series: An overview. *Journal of Contemporary Economic and Business Issues*, 1(1):31–53, 2014.

[117] Y. Ma, W. Gong, and F. Mao. Transfer learning used to analyze the dynamic evolution of the dust aerosol. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 153:119–130, 2015.

[118] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pages 689–696, 2009.

[119] S. Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

[120] S. Mallat. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150203, 2016.

[121] Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *22nd Conference on Learning Theory, COLT 2009*, 2009.

[122] Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation with multiple sources. In *Advances in neural information processing systems*, pages 1041–1048, 2009.

[123] Y. Mansour, M. Mohri, and A. Rostamizadeh. Multiple source adaptation and the rényi divergence. *arXiv preprint arXiv:1205.2628*, 2012.

[124] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

[125] A. Maurer, M. Pontil, and B. Romera-Paredes. Sparse coding for multitask and transfer learning. In *International conference on machine learning*, pages 343–351, 2013.

[126] D. A. McAllester. Some pac-bayesian theorems. *Machine Learning*, 37(3):355–363, 1999.

[127] C. McDiarmid. Concentration. In *Probabilistic methods for algorithmic discrete mathematics*, pages 195–248. Springer, 1998.

[128] A. M. Medina. *Learning Theory and Algorithms for Auctioning and Adaptation Problems*. PhD thesis, New York University, 2015.

[129] L. Minvielle. *Classification d'événements à partir de capteurs sols-Application au suivi de personnes fragiles.* PhD thesis, Université Paris-Saclay, 2020.

[130] M. Moakher. A differential geometric approach to the geometric mean of symmetric positive-definite matrices. *SIAM Journal on Matrix Analysis and Applications*, 26(3):735–747, 2005.

[131] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[132] A. Mueen and E. Keogh. Extracting optimal performance from dynamic time warping. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2129–2130, 2016.

[133] A. Nanopoulos, R. Alcock, and Y. Manolopoulos. Feature-based classification of time-series data. In *Information processing and technology*, pages 49–61. 2001.

[134] G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20:231–252, 2010.

[135] D. Obst, B. Ghattas, S. Claudel, J. Cugliari, Y. Goude, and G. Oppenheim. Textual data for time series forecasting. *arXiv preprint arXiv:1910.12618*, 2019.

[136] E. Ohn-Bar and M. Trivedi. Joint angles similarities and hog2 for action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 465–470, 2013.

[137] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[138] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[139] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.

[140] D. Pardoe and P. Stone. Boosting for regression transfer. In *ICML*, 2010.

[141] O. Parson, S. Ghosh, M. Weal, and A. Rogers. Non-intrusive load monitoring using prior models of general appliance types. 2012.

[142] J. Pearl. *Causality*. Cambridge university press, 2009.

[143] X. Pennec. Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25(1):127, 2006.

[144] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[145] T. K. Pong, P. Tseng, S. Ji, and J. Ye. Trace norm regularization: Reformulations, algorithms, and multi-task learning. *SIAM Journal on Optimization*, 20(6):3465–3489, 2010.

[146] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 262–270, 2012.

[147] J. O. Ramsay. Functional data analysis. *Encyclopedia of Statistical Sciences*, 4, 2004.

[148] I. Redko, A. Habrard, and M. Sebban. Theoretical analysis of domain adaptation with optimal transport. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 737–753. Springer, 2017.

[149] I. Redko, E. Morvant, A. Habrard, M. Sebban, and Y. Bennani. *Advances in domain adaptation theory*. Elsevier, 2019.

[150] G. Richard, B. Grossin, G. Germaine, G. Hébrail, and A. de Moliner. Autoencoder-based time series clustering with energy applications, 2018.

[151] G. Richard, G. Hébrail, M. Mougeot, and N. Vayatis. Densenets for time series classification: towards automation of time series pre-processing with cnns. *MiLeTS@SIGKDD 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[152] G. Richard, A. d. Mathelin, G. Hébrail, M. Mougeot, and N. Vayatis. Unsupervised multi-source domain adaptation for regression. In *Machine Learning and Knowledge Discovery in Databases*, pages 395–411. Springer International Publishing, 2021.

[153] P. Rodrigues, M. Congedo, and C. Jutten. Dimensionality transcending: a method for merging bci datasets with different dimensionalities. *IEEE Transactions on Biomedical Engineering*, pages 1–1, 2020.

[154] M. Rojas-Carulla, B. Schölkopf, R. Turner, and J. Peters. Invariant models for causal transfer learning. *The Journal of Machine Learning Research*, 19(1):1309–1342, 2018.

[155] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich. To transfer or not to transfer. In *In NIPS'05 Workshop, Inductive Transfer: 10 Years Later*. Citeseer, 2005.

[156] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pages 59–66. IEEE, 1998.

[157] S. Said, H. Hajri, L. Bombrun, and B. C. Vemuri. Gaussian distributions on riemannian symmetric spaces: statistical learning with structured covariance matrices. *IEEE Transactions on Information Theory*, 64(2): 752–772, 2017.

[158] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3723–3732, 2018.

[159] A. Sandryhaila and J. M. Moura. Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure. *IEEE Signal Processing Magazine*, 31(5):80–90, 2014.

[160] P. Schäfer. The boss is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29(6):1505–1530, 2015.

[161] P. Schäfer and M. Högqvist. Sfa: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In *Proceedings of the 15th international conference on extending database technology*, pages 516–527, 2012.

[162] I. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[163] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1010–1019, 2016.

[164] L. Shi, Y. Zhang, J. Cheng, and H. Lu. Skeleton-based action recognition with multi-stream adaptive graph convolutional networks. *IEEE Transactions on Image Processing*, 29:9532–9545, 2020.

[165] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[166] C. Song, Y. Huang, Y. Huang, N. Jia, and L. Wang. Gaitnet: An end-to-end network for gait based human identification. *Pattern Recognition*, 96:106988, 2019.

[167] S. Sra. A new metric on the manifold of kernel matrices with application to matrix geometric means. In *Advances in neural information processing systems*, pages 144–152, 2012.

[168] M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. von Bünau, and M. Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4):699–746, 2008.

[169] B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pages 443–450. Springer, 2016.

[170] B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation. In *AAAI*, 2016.

[171] S.-L. Sun and H.-L. Shi. Bayesian multi-source domain adaptation. In *2013 International Conference on Machine Learning and Cybernetics*, volume 1, pages 24–28. IEEE, 2013.

[172] T. Tommasi, F. Orabona, and B. Caputo. Safety in numbers: Learning categories from few examples with multi model knowledge transfer. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3081–3088. IEEE, 2010.

[173] J. Townsend, N. Koep, and S. Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *The Journal of Machine Learning Research*, 17(1):4755–4759, 2016.

[174] O. Tuzel, F. Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classification. In *European conference on computer vision*, pages 589–600. Springer, 2006.

[175] 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*, pages 7167–7176, 2017.

[176] V. Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[177] A. Vaswani, S. Bengio, E. Brevdo, F. Chollet, A. N. Gomez, S. Gouws, L. Jones, Ł. Kaiser, N. Kalchbrenner, N. Parmar, et al. Tensor2tensor for neural machine translation. *arXiv preprint arXiv:1803.07416*, 2018.

[178] R. Vemulapalli, F. Arrate, and R. Chellappa. Human action recognition by representing 3d skeletons as points in a lie group. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 588–595, 2014.

[179] C. Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[180] M. Vlachos, J. Lin, E. Keogh, and D. Gunopulos. A wavelet-based anytime algorithm for k-means clustering of time series. In *In proc. workshop on clustering high dimensionality data and its applications*. Citeseer, 2003.

[181] G. T. Walker. On periodicity in series of related terms. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 131(818):518–532, 1931.

[182] L. Wang, J. Zhang, L. Zhou, C. Tang, and W. Li. Beyond covariance: Feature representation with nonlinear kernel matrices. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4570–4578, 2015.

[183] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

[184] Z. Wang, Z. Dai, B. Póczos, and J. Carbonell. Characterizing and avoiding negative transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11293–11302, 2019.

[185] K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big data*, 3(1):9, 2016.

[186] S. N. Wood, Y. Goude, and S. Shaw. Generalized additive models for large data sets. *Journal of the Royal Statistical Society: Series C: Applied Statistics*, pages 139–155, 2015.

[187] J. Yang, R. Yan, and A. G. Hauptmann. Cross-domain video concept detection using adaptive svms. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 188–197, 2007.

[188] Q. Yang, Y. Zhang, W. Dai, and S. J. Pan. *Transfer learning*. Cambridge University Press, 2020.

[189] X. Yang and Y. Tian. Super normal vector for activity recognition using depth sequences. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 804–811, 2014.

[190] L. Ye and E. Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 947–956, 2009.

[191] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[192] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, page 114, 2004.

[193] C. Zhang, M. Zhong, Z. Wang, N. Goddard, and C. Sutton. Sequence-to-point learning with neural networks for nonintrusive load monitoring. *arXiv preprint arXiv:1612.09106*, 2016.

[194] Q. Zhang, Y. N. Wu, and S.-C. Zhu. Interpretable convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8827–8836, 2018.

[195] B. Zhao, L. Stankovic, and V. Stankovic. On a training-less solution for non-intrusive appliance load monitoring using graph signal processing. *IEEE Access*, 4:1784–1799, 2016.

[196] 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*, pages 8559–8570, 2018.

[197] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao. Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*, pages 298–310. Springer, 2014.

[198] M. Zhong, N. Goddard, and C. Sutton. Latent bayesian melding for integrating individual and population models. In *Advances in neural information processing systems*, pages 3618–3626, 2015.

[199] Z.-H. Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[200] A. Zweig and D. Weinshall. Hierarchical regularization cascade for joint learning. In *International Conference on Machine Learning*, pages 37–45, 2013.

université PARIS-SACLAY | ÉCOLE DOCTORALE de mathématiques Hadamard (EDMH)

Titre: Méthodes d'apprentissage statistique de type "Transfer Learning" pour des données temporelles multivariées

Mots clés: apprentissage par transfert, séries temporelles, adaptation de domaine, apprentissage profond

Résumé: Dans ce travail, nous proposons de nouvelles méthodes d'apprentissage par transfert pour l'analyse des séries temporelles. Motivés par des applications dans le domaine de la désagrégation de la consommation électrique des ménages (NILM) et de la surveillance industrielle, nous étudions les approches traditionnelles et montrons la nécessité de méthodes d'apprentissage par transfert spécifiques aux séries temporelles.

Dans une première partie, nous étudions le prétraitement des séries temporelles pour les réseaux de neurones dans l'optique de la consommation des ménages. Nous proposons une méthode qui permet une meilleure robustesse lors de l'apprentissage sur une maison et l'application sur une nouvelle.

Dans une deuxième partie, nous développons un cadre général pour l'adaptation de domaine adverse pour la régression. Nous fournissons des limites d'apprentissage et un algorithme pour l'adaptation de domaine à une ou plusieurs sources. Nous menons des expériences approfondies sur des ensembles de données privés et publics, montrant de meilleures performances que les méthodes d'adaptation de domaine précédentes.

Enfin, nous étudions les méthodes d'apprentissage par transfert pour les séries temporelles multivariées en utilisant l'information de covariance. Nous représentons les séries temporelles multivariées par leur matrice d'autocovariance et développons un cadre d'adaptation de domaine utilisant la géométrie spécifique de ces matrices.

Title: Transfer Learning for Temporal Data

Keywords: transfer learning, time series, domain adaptation, deep learning

Abstract: In this work, we propose novel transfer learning methods for time series analysis. Motivated by applications in household electricity consumption disaggregation (NILM) and industrial monitoring, we investigate traditional tools and show the need for specific transfer learning methods for time series.

In a first part, we investigate time series pre-processing for neural networks with a view on household consumption. We propose a method that allows for better robustness when learning on one house and applying on a new one.

In a second part, we develop a general framework for adversarial domain adaptation for regression. We provide learning bounds and an algorithm for both single and multi-source domain adaptation. We conduct extensive experiments on both private and public datasets, showing better performance than previous domain adaptation methods.

Finally, we investigate transfer learning methods for multivariate time series using covariance information. We represent multivariate time series by their autocovariance matrix and develop a domain adaptation framework using the specific geometry of those matrices.

Maison du doctorat de Université Paris-Saclay
2e étage, aile ouest, École normale supérieure Paris-Saclay
4 avenue des Sciencs
91190 Gif-sur-Yvette, France