Methodology for construction of adaptive models for the simulation of energy consumption in buildings

Mouna Labiadh

THÉSE de DOCTORAT DE L'UNIVERSITÉ DE LYON

Opérée au sein de :

l'Université Claude Bernard Lyon 1

École Doctorale n° 512
Informatique et Mathématiques

Spécialité de doctorat : Informatique

Soutenue publiquement le 09/09/2021, par :

Mouna Labiadh

Méthodologie de Construction de Modèles Adaptatifs pour la Simulation Énergétique des Bâtiments

Abstract

Predictive modeling of energy consumption in buildings is essential for intelligent control and efficient planning of energy networks. A powerful way to perform this task is through machine learning approaches. Relevant buildings operational data are thus a prerequisite, notably when deep learning is used. However, operational data are not always available, such is the case in newly built or newly renovated buildings.

The goal of this dissertation is to address the predictive modeling tasks of building energy consumption when no historical operational data are available for the given target building. To that end, existing data collected from multiple different source buildings are leveraged. This is increasingly relevant with the growth of open data initiatives in the building energy sector.

The main idea is to transfer knowledge across building models. There is little research at the intersection of building energy modeling and knowledge transfer. A main challenge when dealing with multi-source data is that prominent domain shift can exist between the different sources and also between each source and the target. As a first contribution, a two-fold query-adaptive methodology is developed for cross-building predictive modeling. The first process recommends relevant training data to a target building solely by using a minimal contextual description on it (metadata). To enable a task-specific recommendation, a deep similarity learning framework is used. The second process trains multiple predictive models based on recommended training data. These models are combined together using an ensemble learning framework to ensure a robust performance.

A following contribution consists in the implementation of our methodology based on microservices. "Logically independent" workflows are modeled as separate microservices. Building metadata and operational data (time series) collected from different sources are integrated into a unified ontology-based view.

Finally, experimental evaluations of the predictive models factory is performed for the use case of building energy modeling. Results validate its effectiveness and its applicability. Moreover, because of its generic design, the methodology for query-adaptive cross-domain predictive modeling can be re-used for a diverse range of use cases in different fields.

Résumé

La modélisation prédictive au sein des bâtiments est essentielle pour le contrôle intelligent, la coordination et la planification efficaces des réseaux d'énergie. Un moyen puissant de modélisation prédictive utilise l'apprentissage automatique Avoir des données opérationnelles pertinentes des bâtiments est ainsi un prérequis essentiel pour cette tâche, notamment quand l'apprentissage profond est utilisé. Cependant, les données opérationnelles ne sont pas toujours disponibles, tel est le cas des bâtiments nouvellement construits ou nouvellement rénovés.

Cette thèse s'intéresse à la tâche de modélisation prédictive de la consommation énergétique des bâtiments quand aucune donnée historique n'est disponible. Pour cela, des données collectées à partir de plusieurs différents bâtiments sources sont exploitées. Ceci est de plus en plus pertinent compte tenu la croissance des initiatives de données ouvertes dans le secteur d'énergie des bâtiments.

Ainsi, l'idée est de transférer la connaissance entre les modèles de bâtiments. Peu de travaux de recherche sont menés à l'intersection des domaines de modélisation de l'énergie des bâtiments et le transfert d'apprentissage. Le traitement de données multisources constitue un défi important, vu l'écart de concept qui peut exister entre les différentes sources et aussi entre chaque source et le cible. Comme première contribution, une méthodologie adaptative aux requêtes des utilisateurs est développée pour la modélisation prédictive entre bâtiments. Le premier processus est responsable de la recommandation de données d'apprentissage pertinentes vis-à-vis d'un bâtiment cible, seulement en utilisant une description contextuelle minimale sur ce dernier (métadonnées). Pour permettre des recommandations spécifiques à la tâche cible, l'apprentissage profond de métrique de similarité est utilisé. Le second processus est responsable de l'entraînement de plusieurs modèles prédictifs sur les données d'apprentissage recommandées par le processus précédent. Ces modèles sont combinés avec une méthode ensembliste pour assurer une bonne performance.

La contribution suivante consiste à l'implémentation de notre méthodologie en se basant sur les microservices. Les processus "logiquement indépendants" sont, par conséquent, modélisés en tant que microservices séparés. Les métadonnées des bâtiments et leurs données opérationnelles (séries temporelles) recueillies auprès de nombreuses sources sont intégrées au sein d'une vue unifiée et basée sur des ontologies.

Enfin, des évaluations expérimentales de l'usine de modèles prédictifs sont effectuées à la tâche de modélisation énergétique des bâtiments. Les résultats valident son efficacité et son applicabilité. Par ailleurs, vu le caractère générique de sa conception, la méthodologie peut être réutilisée dans d'autres applications dans divers secteurs.

Acknowledgements

First and foremost, I would like to thank my thesis director and research advisors

- Professor Parisa Ghodous for allowing to conduct this research under her direction. I am also grateful for her consistent support and for the freedom she gave me to do this work.
- Catarina Ferreira Da Silva for her advice and encouragement throughout this research, and for her insightful comments and recommendations on this dissertation.
- Christian Obrecht for his enthusiasm for this project and for his valuable suggestions. His motivation and shared expertise were invaluable in steering this research in the right direction.

Also an acknowledgment goes to EDF R&D for contributing in the funding of my work.

Furthermore, I am deeply grateful to all members of the jury for agreeing to read the manuscript and to participate in the defense of this PhD thesis.

Last but not the least, my deep and sincere gratitude to my parents my brothers Moez and Mehdi for their strong unwavering support and trust throughout my PhD study and my life in general.

Thank you all!

Mouna LABIADH

Contents

Chapter 1

General Introduction

1.1 General Context

Predictive modeling of energy consumption in buildings is crucial to define specific strategies for optimizing energy use within buildings. It therefore helps to achieve intelligent control and efficient planning of energy networks. Specifically, in the context of smart buildings, which are a fundamental part of smart cities, it is increasingly prevalent to apply intelligent algorithms to produce predictive models for anticipating and eventually responding effectively to specific events, such as peaks in energy consumption or changing occupancy patterns.

One increasingly common approach to perform predictive modeling of building energy is through machine learning (ML) [225, 36]. As energy data are time series in nature, time series forecasting approaches are used [63]. These approaches are time efficient and ensure an easy integration of buildings in smart environments [9]. However, the performance of machine learning models highly depends on the relevance and the availability of a sufficient amount of building operational data used for training (multiple months to years of data). This is especially true when deep learning is used due to their inherent complexity [216, 144, 175].

In many cases, historical operational data are not available for training, namely in newly built or newly renovated buildings. With improvements in energy efficiency of such buildings, predictive modeling of energy consumption can not anymore be based on their historical pattern. Moreover, in the field of energy assessment of buildings, energy efficiency is usually verified before construction or renovation. In such cases, only contextual information about the future building is available, namely its design, construction, occupancy, etc.

1.2 A Predictive Model Factory for Building Energy

With the increasing availability of open data in many sectors, existing energy consumption data about *other buildings* may be obtained. The main idea of this work is, therefore, to leverage data collected from multiple different source buildings when *no data* are available on a given target building.

An important challenge arises when working with multi-source data, as large domain shift may exist between different sources and also between each source and the target [235]. Consequently, models trained on source-combined data from disparate sources can interfere with one another during the training process and generalize poorly when applied to a *new previously unseen* target domain [283, 207]. This phenomenon is referred to as negative transfer [255]. More precisely, within the context of building energy modeling, energy consumption depends greatly on several contextual factors, such as the building typology (i.e. residential, industrial or commercial), shape, size and age [264]. Combining energy data from very disparate source buildings is consequently counterproductive and will adversely impact the target performance in the target building.

Proposed approaches addressing the aforementioned challenges fall into the broader field of knowledge transfer, and more particularly *domain adaptation* and *domain generalization* sub-fields. Domain adaptation [260, 37, 213] uses labeled source data as well as sparsely labeled or unlabeled target data to build an efficient model for the target domain. Whereas domain generalization (DG) [286, 28, 179] requires no data in the target domain and uses multiple source domains.

Our work lies somewhere between multi-source domain adaptation and domain generalization areas of research. We aim to train accurate predictive models that perform well on *new previously unseen* target domains via *source domain recommendation*. Source domain recommendation allows to select *most similar* sources to the target domain. Hence, we suppose that *target metadata*, which consist in a contextual description of the target domain, are available beforehand to guide the similarity-based selection process. This contextual description provides a minimal a priori knowledge about the unseen target building, and mainly consists of design properties that are *easy to access*. Conventionally, similarity measures that are used for source domains selection are based on measuring the discrepancy between the distributions of source domains data and target domain data [74, 44, 214]. However, this is not suitable in our case due to the *unavailability* of target domain data during training.

In this work, we aim to learn a metric space, in which source domain recommendation is achieved by computing distances between learned representations of each building. Hence, similarity learning techniques are leveraged to learn building-level representations, so that similar buildings with similar energy consumption profiles are mapped close to each other in the learned feature space, and dissimilar building pairs are mapped far from each other. Representations are generated by training a Siamese network [40, 172] on buildings' contextual descriptions (metadata). In other words, distance between learned domains representations will reflect similarity between their corresponding data, all while requiring only the domains metadata to compute during test. Similarity learning has been proposed in the context of domain adaptation in [195]. However, they assume that unlabeled target domain data are accessible during training. Selecting most similar source domains with the use of target domain metadata rather than actual data via metric learning, as in our work, has yet to be explored.

For the experimental study, we perform predictive modeling of a target building using similar source buildings data. Machine learning based predictive modeling of buildings energy is performed using time series forecasting techniques. As such, a wide range of techniques exists [123, 225, 216], namely artificial neural networks [80, 27] and support vector machines [149, 116]. Conventionally, energy modeling requires many months to years of data on the corresponding building, such as historical consumption data, weather data and occupancy profiles, in order to build an accurate predictive model. A predictive model is, therefore, trained and evaluated on the same building context. Our approach goes beyond such methods and proposes to transfer knowledge across similar buildings. Arguably, there is no single predictive model that works best for every test scenario. This is known as the *no free lunch* theorem [263]. Hence, we use an ensemble learning framework for the predictive modeling task. This allows to combine predictions from several learning algorithms. Ensemble learning generally yields better generalization performance than only one contributing model. As this work aims to provide a generic framework to allow predictive energy modeling for a diverse range of target scenarios, we develop a query-adaptive workflow. Thus, we model metadata on target buildings as queries.

An efficient implementation of the above-described predictive models factory rely on data collected from multiple heterogeneous sources. To provide a unified global view of data, we follow an ontology-based data integration [143, 96]. Defined ontology allows to capture our specific domain knowledge in a formal way. Furthermore, we seek to design different processes of our methodology so that they are sufficiently generic, modular and adaptable. This will allow an easy reuse in similar use cases in other fields, namely finances and healthcare. Therefore, we opt for a microservice-based architecture [72]

1.3 Contributions

This PhD research has the following contributions :

- To our best knowledge, this work is a first attempt to transfer knowledge across buildings with no historical data available about the target building during training. To that end, a novel methodology is developed. Two levels of information about buildings are leveraged, namely contextual description about the building (metadata) and historical data. As such, to mitigate historical data unavailability for a given target building, metadata (easy to acquire) about it are used instead.

  Firstly, most similar buildings from available source buildings are selected based on task-specific inter-domain similarity information. Historical data from these buildings are subsequently retrieved and recommended as training data. An appropriate similarity measure is learned for this task using a deep similarity metric learning framework. Secondly, predictive models are trained using recommended data. To improve robustness and accuracy, an ensemble learning framework is used. The experimental evaluations validate the effectiveness and the applicability of this methodology.

Given the generic nature of its design, our methodology can be re-used for different use cases in various sectors.

- A microservice-based implementation of cross-building energy modeling is designed. Logically independent tasks are consequently designed as individual microservices. In the context of this work, we propose to collect building energy-related data from different data sources, such as open data and simulation results. For efficient integration of such data, a unified global view is used. More particularly, metadata about buildings (contextual descriptions) are integrated using ontologies. We extend ThinkHome ontology [205] to describe building design and thermal parameters, and basic information about occupants. Historical time series data are, on the other hand, integrated in specific data stores and referenced from corresponding building's metadata view.

1.4 Outline of the thesis

The rest of this report is structured as follows. The first part provides an overview of the necessary concepts to understand the research aspects of this PhD work. It also provides a review of related literature used to formulate the research proposal. Chapter 2 presents the general use case of building energy modeling. Chapter 3 provides a brief background on machine learning and deep learning, and an in-depth background on various time series forecasting techniques. Chapter 4 focuses on various approaches for cross-domain knowledge transfer. The second part details the main contributions of this PhD research work. Chapter 5 details the proposed methodology and its main components. Chapter 6 presents the proposed microservice-based implementation and data specification. Chapter 7 details the experimental setup and discusses experimental findings. Finally, in Chapter 8, we conclude with possible directions for future work. Figure 1.1 illustrates the outline of this manuscript.

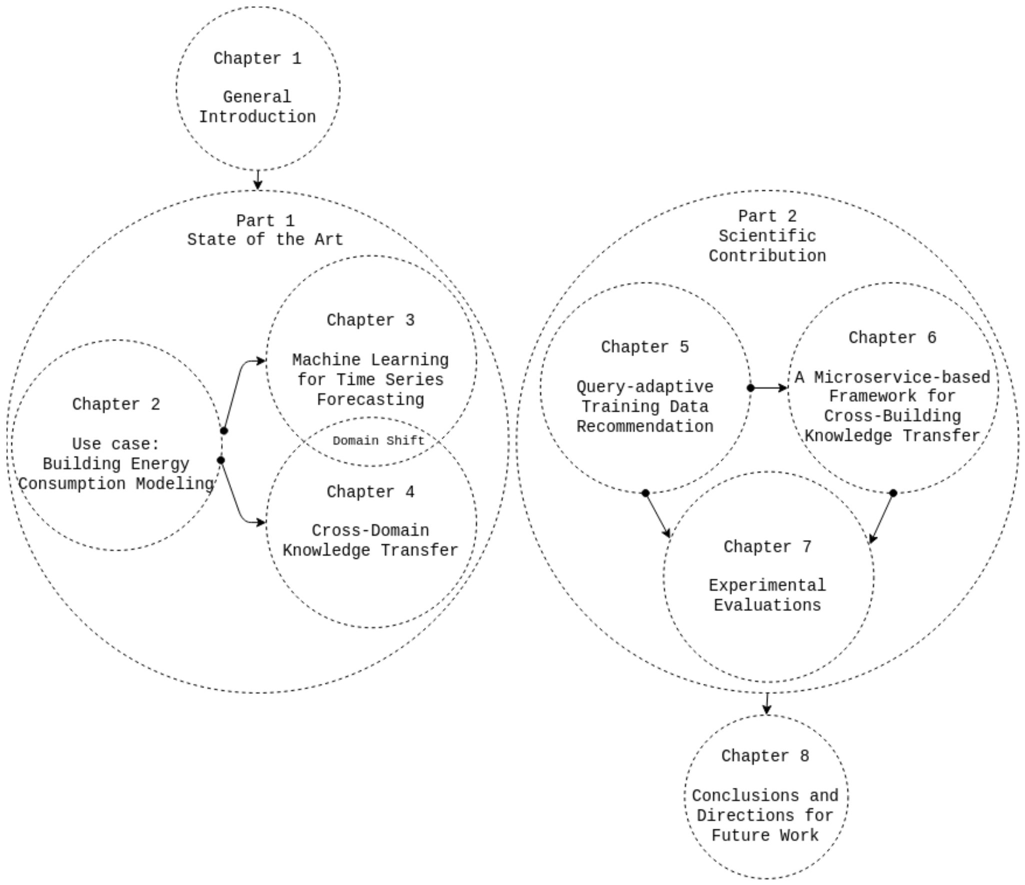

Chapter 1
General Introduction
Part 1
State of the Art
Chapter 2
Use case: Building Energy Consumption Modeling
Chapter 3
Machine Learning for Time Series Forecasting
Domain Shift
Chapter 4
Cross-Domain Knowledge Transfer
Part 2
Scientific Contribution
Chapter 5
Query-adaptive Training Data Recommendation
Chapter 6
A Microservice-based Framework for Cross-Building Knowledge Transfer
Chapter 7
Experimental Evaluations
Chapter 8
Conclusions and Directions for Future Work

Figure 1.1: Outline of the manuscript structure.

Part 1

State of the Art

Chapter 2

Building Energy Consumption Modeling

2.1 Introduction

Buildings account for over one-third of the global energy consumption as for recent years, and nearly 40% of greenhouse gas emissions. As population and floor area growth continues, global energy demand in buildings is rapidly increasing. Consequently, improving the energy efficiency of buildings has become a key issue of concern in innovative research, construction industry, and government programs and policies.

Several efforts and initiatives are carried out to improve building-level energy efficiency by helping to measure, monitor, optimize and evaluate energy-related building services such as heating, ventilation, air conditioning and lightning. Optimal management and conservation of building energy rely on accurate prediction of future energy consumption. This is assured by building energy modeling (BEM) tools. In addition, when evaluating building energy use, BEM is used for benchmarking and rating, by measuring energy performance of an individual building over time and compare it to other similar buildings or reference buildings. BEM is therefore a multi-purpose module that is employed for various cases, including assessment of new buildings design, renovation and retrofit of existing building, real-time management and control of energy consumption and utility incentives.

This chapter is structured as follows. Section 2.2 provides a brief overview on building energy modeling, including its common uses and its taxonomy. Section 2.3 provides a deeper view into machine learning based approaches as well as their limitations. Section 2.4 broadly presents determinants of energy consumption in buildings, and reviews existing information models. Finally, in Section 2.5, we draw some conclusions.

2.2 Building Energy Modeling

Mathematical modeling is an approach to describe complex systems using mathematical concepts in order to help understand them and make predictions about their behavior. As such, building energy modeling involves the development of models that can capture key properties of a system - in this case building energy consumption - where subsequently simulations can be performed with different sets of input data to study future changes [222].

Building energy modeling (BEM) plays an important role for the efficient planning and operation of energy networks. It mainly consists in predicting future energy consumption. As such, an accurate prediction of energy consumption at the building level will promote the building energy utilization rate, and will directly improve the efficiency of the planning process of the entire energy network [175]. Despite the long-standing interest in BEM, it remains as a challenging problem given the wide range of factors that may affect the energy use in a building, namely economic and socio-demographic factors [84, 50, 87], weather conditions, occupants' behaviors [65], physical building characteristics, operation of sub-level components such as HVAC systems (heating, ventilation and air-conditioning), lighting systems and electric appliances [282].

2.2.1 Common use cases

Building energy modeling has multiple uses. It can be used in the context of energy performance assessment at building-level. This is generally performed for new buildings at its design process, or for existing buildings at its renovation process. In fact, a building life-cycle is composed of three main stages: design stage, construction stage and operation and maintenance stage (see Figure 2.1). Design stage involves a detailed planning of building space layout, envelope, etc. There are several aspects required to take into account such as energy efficiency, environmental and health impacts, life-cycle costs and urban integration. Energy efficient buildings are designed to ensure minimum energy demand. Once constructed, a building enters its operation stage (also called occupancy stage). Operational energy is important during this stage, to provide lighting, heating, ventilation, air conditioning, power control systems, and appliances. During its operation, a building may undergo many changes in its design and function. Changes ranges from minor such as replacing appliances (remodeling) to more significant such as expanding the building or upgrading the thermal envelope for improved energy efficiency (renovation). To conclude, BEM is used to study and benchmark the overall building performance of an existing or a new building in order to establish construction or energy renovation strategies. As any product, buildings have finite lives. End-of-life stage thus consists in demolition [58].

BEM is also generally used in the context of urban energy modeling. Urban energy modeling aims to better understand energy consumption in new or existing districts, cities, etc. This allows an effective planning of energy retrofitting programs or changes

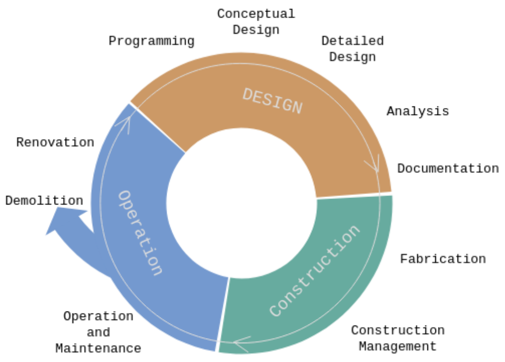


Figure 2.1: Broad view of building life-cycle.

in energy supply infrastructure. For instance, urban energy modeling techniques can be distinguished into two main categories (1) Top-down approaches and (2) bottom-up approaches. *Top-down approaches* rely on load data profiles of the entire district or city, and then it is increasingly down-scaled into smaller sub-sections until reaching building-level energy consumption. By contrast, *bottom-up approaches* use information on the building stock and predict the energy consumption based on the current and predicted characteristics of the building stock itself. As such, BEM is commonly used for bottom-up approach to build archetypes that represent "typical" building stock's categories.

More recently, BEM is increasingly prevalent in the context of smart buildings, which are a fundamental part of smart cities. A smart building is controlled by a building management system (BMS) which gathers data from sensors and meters. BMS therefore relies on energy data to dynamically determine in real time the most suitable decisions to undertake in order to improve the building energy efficiency, such as peak clipping, load shifting and valley filling. Peak clipping/shaving includes introduced techniques to reduce peak power demands on the system. Load shifting involves shifting power consumption to another time period, typically during off-peak hours and when energy prices are lower. Valley filling a power consumption profile consists on encouraging additional energy use during periods of lowest system demand. Therefore, BEM is crucial for forecasting building-level power loads and peak loads. It allows to accurately and dynamically identifying future energy consumption and energy saving opportunities for building energy management systems.

2.2.2 Taxonomy of Building Energy Modeling Approaches

Mathematical modeling encompasses a variety of techniques including physical principles, applied statistics and machine learning. Physics-based modeling is application-domain specific as it largely depends on physical relationships within the studied system. While statistical and machine learning modeling is data-driven. Some hybrid approaches that lies between both axis also exist. More precisely, mathematical modeling of building energy approaches can be distinguished into three approaches [36, 8, 282]: (1) white-box based, (2) grey-box based and (3) black-box based. *White-box based models* (also known as physics-based models or engineering-based models) [8] consist in providing a simplified explicit representation of the building components using physical equations, and resolving it. These models require a detailed description of the geometrical structure and thermo-physical characteristics of the building, and the environmental conditions. A lot of specialized and mature white-box software tools exist, such as EnergyPlus [59], ESP-r [244], eQuest and TRNSYS [247]. A white-box based approach allows the creation of flexible and transparent benchmarking models for the assessment of the effectiveness of design alternatives of new buildings and the potential of deep energy retrofits of older buildings. However, these models are computationally expensive and require a considerable amount of time and expertise for the identification of detailed information about the building phenomena, and the calibration of the dedicated simulation engine for each building [253]. This makes this approach not amenable to scale to the urban level and unsuitable for online or near real-time applications. Studies also find that white-box models may inaccurately predict energy use in real-world operational buildings given their high complexity [46]. As such, white-box models may fail to account for the stochastic occupant energy use behavior [218], complexity of the built environment, uncertain inputs, and inefficiencies in operational buildings due to improper maintenance and operation of building systems [170].

The *grey-box based models* (also known as hybrid models) [119, 126] are a modification of the white-box based models through the use of statistical methods. They use a simplified physical description of the building energy systems. An unknown subset of the model parameters are then estimated using historical data through statistical algorithms. Even though grey-box approach has some advantages, such as low calculation time and flexibility, it also combines limitations of the white-and-black-box based approaches, such as the computational inefficiency and the failure to account for the complex interactions between building elements.

The *black-box based models* (also known as data-driven models) [216, 8] use historical data to model the energy consumption of buildings, instead of detailed physical descriptions. We distinguish between *statistical* and *artificial intelligence* methods [216]. Statistical methods use historical data to find a correlation between the most influential features as input and the energy consumption. Instances of statistical models include auto-regressive moving average (ARMA) [182], conditional demand analysis (CDA) [150], Gaussian mixture models (GMM) [225], etc. Artificial intelligence (AI) methods use historical data to model the behavior of the process to be modeled. Instances of AI models include artificial neural networks (ANN) [80, 27], support vector machines (SVM) [149, 116] and decision trees [275]. The scope of this research is focused on the data-driven approaches, and more precisely machine learning based techniques.

We summarize pros and cons of white-box, grey-box and black-box approaches in Table 2.1.

Table 2.1: A summary of the pros and cons of three time series forecasting techniques

Approach	Pros	Cons
White-box	• Based on detailed description of the building physics • Transparent • No need for training	• Computationally expensive and time consuming • Difficult to model real scenarios • Generally require oversimplifying assumptions
Grey-box	• Based on simplified physical description of the building	• Computationally expensive • Fail to model real scenarios
Black-box	• Deep understanding of underlying principles not required • Fast computation • Suitable for real-time data processing • Suitable for non-linear modeling • Easy integration in smart environments • Generally more accurate than white-box model	• Require large amount of historical data • May not be transparent (deep learning)

2.3 Machine Learning for Building Energy Modeling

Machine learning (ML) is a subset of artificial intelligence that provides the ability to automatically learn and improve using data. One main advantage of using ML-based models, or data-driven models in general, is that they allow a facilitated integration into smart systems. Consequently, it is increasingly prevalent to apply machine learning algorithms to produce predictive models. Smart buildings generally rely on an automated energy management infrastructure that include smart meters, sensors and the communication network. Smart meters continuously collect data such as energy consumption, while sensors gather environmental data such as motion, temperature or lighting. As such, building energy-related data are collected are time series. Technologies like internet-of-things (IoT) are used to simplify tasks like building control and to effectively manage building devices.

Works on BEM in literature may be distinguished depending on several factors [8], such as:

- Type of building, including residential, commercial or office buildings.
- Time-resolution of predictions, including short-term (e.g. sub-hourly, hourly, daily) or long-term (e.g. yearly).
- Input features, including historical consumption data, weather data and building design parameters.
- Dataset size as the number of timesteps used for training and testing the ML model.
- Machine learning algorithm, including support vector machines, artificial neural networks and decision trees [63, 225, 36].
- Target variables, including whole-building level energy consumption (total) or sub-level components, such as cooling, heating or lighting.

Several reviews on machine learning-based BEM approaches and their classification are proposed in literature [63, 8].

Most commonly-used machine learning algorithms for BEM are support vector machines (SVM) [149, 229, 116, 123, 175] and artificial neural networks (ANN) [22, 80, 27, 175]. Jain et al. [116] studied a SVM with a radial basis function (RBF) kernel for predicting electricity consumption in a multi-family residential building, based on different temporal granularity (daily, hourly, every 10min) and different spatial granularity (whole building, by floor, by unit). Two datasets are used, that contain data over respectively 6 months and 3.5 months. As input features, authors used historical consumption data, current outdoor temperature data, current solar flux data, weekend/holiday or weekday indicator, sine of current hour, and cosine of current hour. Zhao et al. [281] proposed a parallel implementation of SVM with RBF kernel to predict hourly heating consumption and hourly electricity consumption for multiple office buildings. Energy data were simulated using EnergyPlus. Number of buildings in the dataset is 100 and each building has hourly data for over 5 months. Input features are historical data, holiday indicator, weather conditions, zone mean air temperatures, infiltration volume, heat gain through each window, heat gain through lights and people and zone internal total heat gain.

Biswas et al. [27] proposed an ANN-based models to predict daily total and heat pump energy consumption in an unoccupied residential building. Input features are number of days, outdoor dry-bulb temperature and solar radiation. The dataset spans 3 months. Ben-Nakhi et al. [22] proposed a general regression neural networks (GRNN) to predict hourly cooling loads in office buildings. The experimental dataset was generated using ESP-r and contains 3 buildings. Corresponding data cover a time span of 5 years. Input features consist only on previous day's external dry-bulb temperature. Outputs consist in hourly cooling loads of subsequent day.

In recent years, deep learning is widely adopted for energy modeling tasks, and becomes the state of the art when large amount of historical data are available. Various deep learning models have been used, such as recurrent neural networks (RNNs) [175, 168, 129, 252, 130], or a combination of convolutional neural network (CNN) and RNN (CNN-RNN) [245, 127]. Kong et al. [130] proposed a long short-term memory (LSTM) RNN model for load forecasting in residential buildings. The proposed model was compared to conventional backpropagation neural network, k-nearest neighbors and extreme learning machine. Dataset spans over 3 months and contains 69 buildings. Data are recorded every 30 minutes. Input features are the sequences for past K time steps ($K = 2, 6, 12$) of historical consumption data, day time indices, day of week indices and binary holiday marks. Output is the energy consumption forecast for the target time interval. Kim et al. [127] proposed a CNN-RNN model for predicting minutely electric power consumption in a residential building. Dataset spans over 4 years. CNN-RNN model takes as input a 60-min window and provides as output the predictions of the next 60 minutes.

A detailed explanation of machine learning notion and the most common algorithms discussed above is found in Chapter 3.

Despite their success in the context of energy modeling, machine learning approaches have two main limitations. First limitation is that ML requires a sufficiently large amount of data to produce accurate models, notably when deep learning is used. As such, buildings for which no (or few) energy consumption data are available (e.g. newly built buildings or newly renovated buildings) can not be successfully modeled.

Secondly, and more importantly, machine learning models trained on a source distribution (training dataset) may perform poorly when used in the context of a different target distribution. For instance, a model trained on building A may not generalize well on a different building B (e.g. different locations, different types of buildings, different design, etc.). Similarly, a model that predicts energy consumption in a building, may become unusable when this building changes over time. Changes in energy consumption are generally attributed to renovation works, changes in occupancy (e.g. change of tenants), changes in operations schedule, etc. As such, a different predictive model is generally trained for each building, and a predictive model of a particular building is often required to be re-trained to be up-to-date. Alternatively, solutions for training models that generalize well across different data distributions are proposed in the framework of *knowledge transfer*. A detailed presentation of this field and its common techniques can be found in Chapter 4.

2.4 Building Energy Data

Data-driven approaches in general, and machine learning based approaches in particular, typically require a sufficient amount of data on several exogenous variables. These determinant variables of energy consumption are numerous and diverse. It is therefore required to identify most relevant variables for accurate BEM.

2.4.1 Determinants of Energy Consumption

Buildings are complex systems that must ensure the comfort of occupants, and therefore must provide heating, ventilation and possibly air conditioning, which are responsible for a significant proportion of the total energy consumption within buildings.

Building Envelope

To model the energy demand, it is necessary to take into account heat and mass transfer phenomena. Heat transfer occurs via three mechanisms: conduction, convection and radiation [136]. Conduction occurs within any material body through propagation of thermal energy in case of temperature gradients. In buildings, heat conduction occurs through the envelope in response to weather fluctuations, solar radiations and ground temperature, as well as through partition walls, indoor floors and ceilings. Building heat conduction problems are generally multi-dimensional and transient. However, for the sake of computational efficiency, one-dimensional multi-layer numerical models are generally used. Such models allow accurate understanding of heat conduction through building envelope under the assumption that heat flow is normal to the surface. A multi-dimensional analysis is more relevant in case of building corners, balconies, and other potential thermal bridges.

Convection, which results from fluid displacement, causes convective exchanges at the interface of solid surfaces. This kind of exchanges through inner and outer surfaces of the building envelope are of major importance in the thermal balance of a building and must be considered. In buildings, convection occurs through inter and intra-surface air motion as result of temperature and pressure differences, or through outer envelope due to air infiltration.

Radiative heat transfer is caused by the thermal electromagnetic emission of material bodies with respect to their temperature. In building physics, distinction is made between short-wavelength and long-wavelength radiations. Short-wavelength radiation, either direct or diffuse, originates mainly from solar light emissions. In buildings, solar radiation is absorbed by opaque outer surfaces, transmitted by transparent or translucent surfaces, and eventually absorbed and reflected by interior surfaces and glazed openings. Long-wavelength heat exchanges are, on the other hand, a characteristic of heat exchange between building components. In buildings, long-wavelength heat is emitted from exterior surfaces to the sky, the ground and other surrounding objects, exchanged among interior walls, as well as between interior walls and occupants.

Building thermal balance relies on several factors related to building geometry, materials, occupancy, weather, etc. For example, on a sunny day, solar radiation passes through glazed openings (windows or doors), resulting in heat gain within the inner volume. Heat gain depends on the transmission rate of the glazing and on the orientation of the openings, hence the need to have a notion of building orientation. Moreover, heat from solar radiation will also contribute to the energy balance by conduction through the outer wall to the inside wall surface from which it is convected and radiated into the building.

The thermo-physical properties of the building envelope is of major importance in determining the indoor air temperature, and thus the energy demand for heating and cooling. Therefore, information about construction material, dimensions and other building parameters is of high value. Fundamental thermo-physical properties of building parameters are thermal conductivity, density and heat capacity. Information about occupancy and equipment profiles and schedules is also fundamental. For instance, human body generates sensible and latent heat through metabolism, and dissipates it by radiation, convection, and evaporation.

Occupants Behavior and Activities

Occupants behavior and activities greatly influence energy consumption within buildings in several manners [65]. Occupants continuously interact with building systems to control indoor environment and ensure comfort. This includes indoor air quality (e.g. circulate fresh air and eliminate indoor air pollution), thermal comfort (e.g. control indoor temperature set-points for heating and air conditioning), visual comfort (e.g. control view, luminance ratios and avoid screen reflections) and acoustic comfort (e.g. minimize intruding noise and vibrations) [30].

Socio-economic and Demographic Factors

Socio-economic and demographic aspects also play a role in impacting energy consumption. Demographics factors may include age of occupants and level of urbanization, [98]. Economic factors include energy prices, pricing schemes and household revenue. Namely, building adopting time-of-use pricing will modify their energy consumption patterns voluntarily to reduce their energy expenses. Other factors are time spent at building, culture, social status, and education and awareness level of occupants. For instance, according to some researchers, the higher is the occupants education level, the more likely they are to engage in saving energy [25].

Outdoor Conditions

Moreover, outdoor conditions, such as surrounding urban morphology and environmental factors like air temperature, wind and humidity, highly impact energy consumption. High air drought induce more pressure on the building wind-ward walls, which leads to higher indoor air change. Air infiltration also occur due to differences between indoor and outdoor temperature.

To summarize, energy consumption in buildings is often complex as it depends on many determinant variables. Consequently, several parameters are required to perform building energy modeling. Such parameters broadly consist of its corresponding geometric shape, orientation, physical properties of the building envelope and interior elements, occupation scenarios, and systems. Information about meteorological factors, such as wind speed, relative humidity, air temperature, and solar irradiance is also often required. Figure 2.2 provides an overview on some determinant variables of energy consumption within buildings.

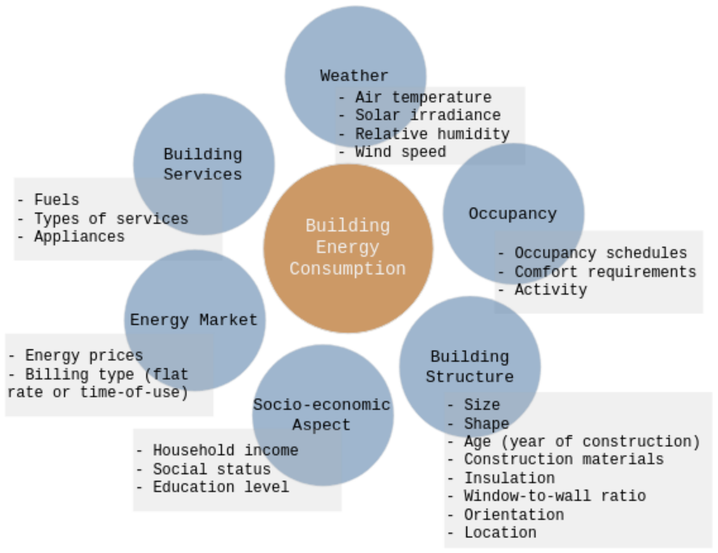


Figure 2.2: Categorization of building energy data on determinant variables.

As such, two forms of data are required for an accurate building energy modeling, namely time series data and contextual descriptions. Time series data represent variables whose values follow a temporal ordering, such as historical energy consumption and meteorological conditions. Contextual descriptions, on the other hand, represent static features of the building, such as type, design, construction, etc. We refer to contextual descriptions of buildings as *metadata* for the rest of this work.

2.4.2 Metadata Schemas for Building Energy Modeling

Building energy modeling require detailed information about the building design and operation parameters. Parameter instances are structured in information models. Traditional pipeline goes from a 2D or 3D building model that is realized with Computer-Aided Design (CAD) tools (i.e. REVIT and SketchUp). Data are then sequentially exchanged between actors throughout the building life-cycle. At each exchange, data are translated into suitable formats for different applications. Such approach leads to numerous issues as it lacks coordination between building professionals, it fails to provide a common data platform for distributed organizations and among concurrent decision processes, and it induces inefficient project management and costs estimation. Building Information Models (BIM) technology was therefore introduced to establish synergistic workflows that link the Architecture, Engineering and Construction (AEC) industry. BIM achieves a higher level of interoperability which enables data sharing without any loss information throughout the building life-cycle. BIM offers a richer representation of the building through multidimensional models [3]: 3D is a visualization model, 4D is the time monitoring, 5D is the costs estimation, 6D is the energetic analyses and the 7D correspond to the facility management of a building. Data interoperability includes the ability to exchange and accurately interpret information across heterogeneous applications. Interoperability consists on transforming each application's internal data into an universal model and vice versa through data mapping [185]. Consequently, great attention was paid to standardize BIM processes.

Over the years, several data models have reached a high level of maturity in supporting the whole building life-cycle. First proposed building data exchange models were STEP [261, 198], ARMILLA [100], IBDE [79], ICADS [196], which are mainly interested on building geometrical information and therefore not sufficient. Later generation of data models sought to include domain-specific processes, such as building energy simulation, thermal comfort evaluation, building design evaluation, project planning, etc. The hierarchical model [19] focuses on life-cycle cost modeling. KNODES [217] further covers building energy modeling. SEMPER-II [141] proposes a platform that supports integrated concurrent design support. The last generation of data models offers the potential to encapsulate all information related to the built environment, yielding the BIM paradigm.

Currently, two prevalent data exchange formats are used in AEC industry; Industry Foundation Classes (IFC) developed by buildingSMART and Green Building XML (gbXML) [88] developed by Green Building Studio. IFC represents a major effort to standardize AEC data, and is widely accepted as a mature and comprehensive model within the industry. IFC is based on ISO STEP data modeling standards. Data are thus specified in EXPRESS modeling language. Its goal is to provide a platform-neutral open data format that enable interoperability across BIM applications. The IFC schema defines objects and relationships that cover core building information such as building elements, the geometry and construction material properties, cost management, construction scheduling, and organizations.

A gbXML model captures geometric information (e.g. walls, shading surface and windows) and non-geometric information including occupancy, equipment, lighting, HVAC (heating, ventilation and air-conditioning) system set points, and operation schedules. Because gbXML is based on XML, it is structured as a tree of elements with attributes as shown in Figure 2.3. Geometry information is contained in "Campus" element. The child element "Surface" represents all surfaces in the geometry. Walls, ceilings and floors are usually modeled as planar surfaces. A planar surface is defined by four boundary points given their Cartesian coordinates. Openings (e.g. windows, doors) are modeled as rectangles co-planar to the walls to which they belong. Semantic information such as construction, layer and material is related to its corresponding geometry using reference attributes. A construction consists of a building material composite. A building material composite consists of corresponding layers. Material layers are described by their physical and thermal parameters (e.g. thickness, conductivity, density). Alternatively, material layers representation can be simplified as one layer representing the thickness and the U-value of the composite. U-value expresses the thermal transmittance of the composite.

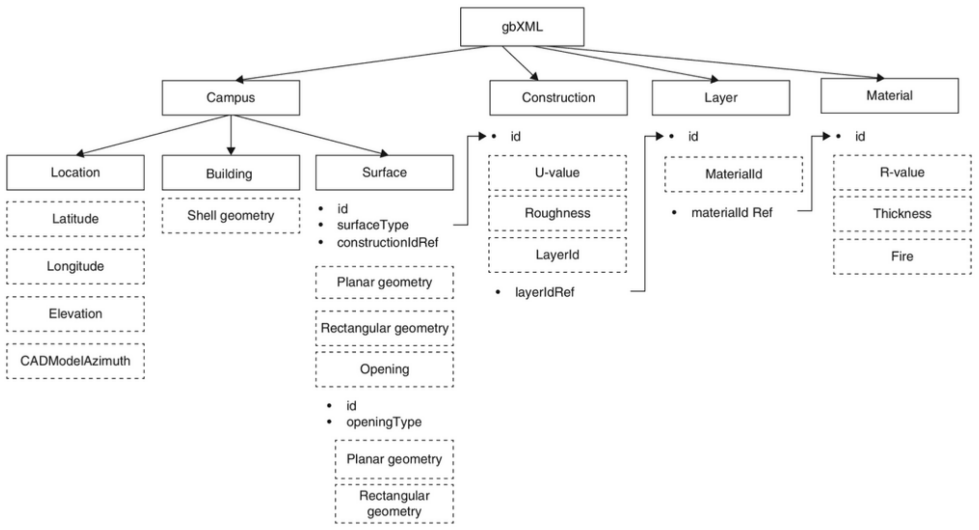


Figure 2.3: Graphical visualization of a subset of gbXML model elements [10]. gbXML elements are represented with solid rectangles, while their attributes are represented with dashed rectangles. Arrows represent parent/child relationships among elements.

However, IFC and gbXML do not ensure semantic interoperability. For instance, the IFC does not allow the specifications of all elements required to express HVAC systems. To achieve semantic interoperability, semantic data models were proposed e.g. BOT [204], SAREF4BLDG [197], ThinkHome [205], DogOnt [31], ifcOWL [193]. Ontologies seeks to add a richer semantic layer on top of objects, parameters definitions and relationships defined in other schemas, as well as reasoning and inferring possibilities. Other models extend and focus on other aspects of the built environment such as the IoT infrastructure [200][14].

2.5 Summary and conclusions

Building energy modeling is a versatile and multi-purpose module that enables designers, end users, energy suppliers and policymakers to model buildings energy usage in both new buildings during design and existing buildings during renovation and retrofits. BEM is also used in the context of smart buildings to promote intelligent energy management. BEM is essential to promote energy efficiency within buildings and effective energy planning at the whole energy network.

In this chapter, a categorization and a comparison are established of different BEM approaches, namely white-box, black-box and grey-box based. In particular, BEM using machine learning, which is a subset of the black-box based modeling, is increasingly prevalent given its numerous advantages. In addition to its accuracy, this approach is time and computationally efficient, and allows an easy integration of buildings into smart systems. An introduction to machine learning, and a detailed review of techniques that may be used in the context of BEM are found in Chapter 3.

We categorize ML-based approaches reported in literature, depending on learning algorithms, time resolution of predictions, target variables, etc. Building energy consumption data, being time series in nature, are modeled commonly using support vector machines and artificial neural networks. In addition to historical energy consumption, several exogenous variables may influence consumption patterns in buildings, namely thermo-physical properties of building envelope, occupancy, weather condition and socio-economic factors. Data on these exogenous variables provide contextual description about building energy, and therefore considered as *metadata* for BEM. We overview commonly proposed schemas for the integration of buildings metadata, namely IFC and gbXML. Moreover, several ontologies are proposed in the context of BEM, such as ThinkHome. Ontologies add a layer of semantics that delivers a common and formal description of knowledge.

Despite their success, accurate ML models generally rely on large amount of data. This limits its usability in cases when no or few historical data are available for training, such is the case in newly built or newly renovated buildings. In addition, models that were trained on data from source building may generalize poorly when used in the context of a different target building. As such, historical data unavailability for a given building can not be always overcome by simply training a model on a different building (on which enough data are available). This is also frequently the case of individual building that have undergone renovation works, which results in changes in energy consumption profiles. Example of solutions that were proposed to overcome such challenge are made in the context of *Knowledge transfer*. A deeper view of this field of research and proposed techniques is found in Chapter 4.

Chapter 3

Machine Learning for Time Series Forecasting

3.1 Introduction

One powerful way to perform building energy modeling is through machine learning. Machine learning represents a profound paradigm shift in programming. Traditional programming refers to manually create deterministic algorithms, that detail every step that should be taken to perform a specific task. Traditional programs thus takes data as input and process them as stated by the rules. When data do not fit within explicit rules, they are unable to process them. Similarly, when rules are too complex for a human to explicitly and accurately program, traditional programming is not appropriate. By contrast, machine learning refers to training a model that learn from data themselves without the need of an explicit program. Figure 3.1 illustrates the difference between traditional programming and machine learning.

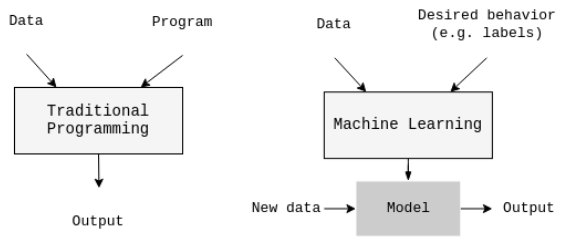


Figure 3.1: Traditional programming vs machine learning.

Machine learning based building energy modeling aims to accurately predict future energy use in buildings. Energy use data are time series in nature. Hence, most applied ML techniques in this context are artificial neural networks and support vector machines. More recently, deep learning is increasingly adopted and becomes the state-of-the-art approach when large amount of historical data are available.

This chapter is structured as follows. Section 3.2 overviews the taxonomy of machine learning approaches, and details the classic machine learning workflow. Section 3.4 presents techniques that are proposed for time series forecasting. Section 3.5 concludes this chapter.

3.2 Learning from Data

Machine learning is a subset of artificial intelligence that uses statistical decision theory to build systems that have the ability to automatically learn from data. In [174], a definition of learning is provided: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". In general, a well-posed learning problem relies on the identification of these three features: class of tasks, measure of performance to be improved, and source of experience.

A task is different from the learning itself. The learning is simply the process of attaining the ability to perform the task [92]. A performance measure is a quantitative measure that is used to evaluate the machine learning algorithm. The choice of a performance measure depends mainly on the task carried out by the system. An experience refers to the past information available to the learner, which typically consists of data collected for analysis. The amount and the quality of data are crucial to the success of learning algorithms. As an example, a robot that learns to drive in a four-lane highway using visual sensors, might improve its performance as measure of average distance traveled before an error, through experience obtained by a sequence of images and steering commands recorded while observing a human driver (example taken from [174]).

Data used for machine learning forms a *dataset*. A dataset consists of data instances called *examples*, *data points* or *inputs*. An example is a collection of *features* that are quantitatively measured of a particular observation of the phenomenon that the machine learning system is designed to process. Typically, an example is represented as a vector $\mathbf{x} \in \mathbb{R}^n$, where each entry x_i is a feature. A common way of representing a dataset is with *design matrix*. A design matrix is a matrix where each row corresponds to an example and each column corresponds to a feature. A machine learning task depends on the way we are processing an example. Typical examples of tasks are classification and regression. For classification tasks, the computer program maps examples to categories they belong to. For example, in email classification, each email is mapped to either spam or not spam categories. We therefore seek to learn a target function $f : \mathbb{R}^n \mapsto 1, \ldots, K$, where $1, \ldots, K$ are numerical codes for the categories. In a regression task, the computer program predicts numerical values given input examples. The unknown target function is $f : \mathbb{R}^n \mapsto \mathbb{R}$. An example of a regression task is the prediction of sale prices of houses [6].

3.2.1 Taxonomy of Machine Learning Techniques

The primary premise of machine learning is the use of data to uncover an underlying process [2]. Several learning paradigms have been introduced to address different settings and different assumptions. A common learning paradigm classifies ML techniques based on available information. We distinguish between supervised, unsupervised, and reinforcement learning.

Supervised learning

Supervised learning denotes learning problems where the training data consists of an input and an output (also called label, target or response). The goal of supervised learning is to make label predictions for unlabeled unseen data.

The training dataset $\mathcal{D}$ thus consists of input-output pairs :

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), ..., (\mathbf{x}^{(N)}, y^{(N)}) \subseteq \mathcal{X} \times \mathcal{Y}$$

where N is the number of training examples, $\mathcal{X}$ is the input space (set of all possible inputs $\mathbf{x}$), and $\mathcal{Y}$ is the output space (set of all possible outputs y), y is supposed to be deductible from x by a process set up by learning.

The learning is said "supervised" in the sense that we assume there is a knowledgeable external "supervisor" that will provide outputs. Supervised learning tasks are related to classification or regression.

Unsupervised learning

Unsupervised learning consists in handling data that are not labeled. The goal is to discover underlying patterns and structure in the data. This is also known as *knowledge discovery*. Often, unsupervised learning tasks are related to density estimation, uncovering a manifold structure that data lies near, learning to draw samples from a distribution, learning to denoise data from a distribution, clustering data into groups of similar examples [92]. Unsupervised learning can also be used to extract high-level representations from data. This is particularly useful to learn lower-dimensional data representations, that preserve as much information about the original data to perform the target task.

Reinforcement learning

Reinforcement learning (RL) differs from other ML techniques. RL paradigm concerns control problems, rather than analyzing given data. For example, consider the problem of a robot learning how to navigate or play chess. In reinforcement learning, there is no target output, but instead a set of possible outputs and a measure of how good each output is. The goal of RL is teaching an agent how to behave optimally using rewards and penalties as signals for positive and negative behavior.

In Figure 3.2, we provide a broad classification of various examples of machine learning algorithms.

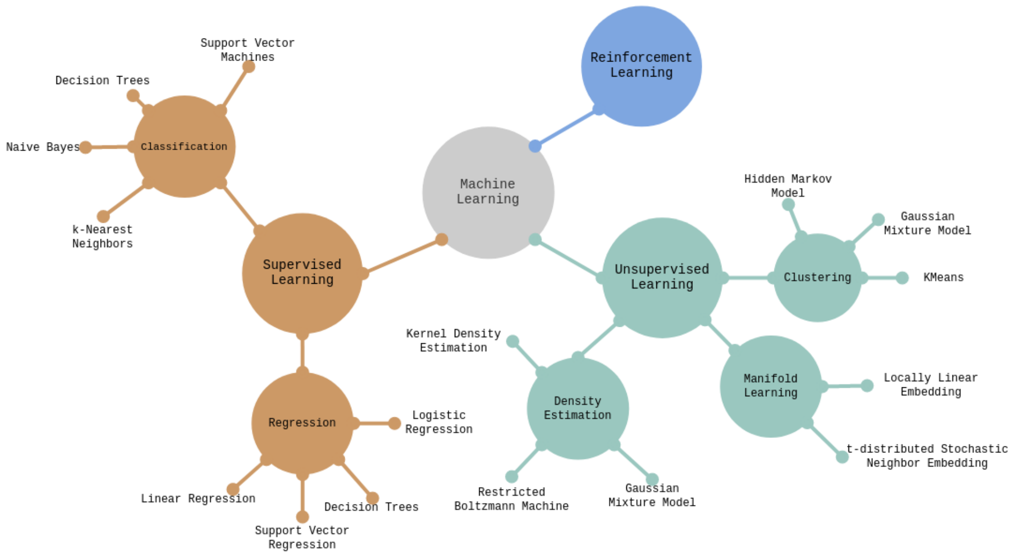


Figure 3.2: Broad Classification of Some Machine Learning Algorithms.

Other learning paradigms

Other learning paradigms exist to distinguish different learning techniques:

- **Passive vs Active :** Passive learning consists in standard supervised learning, where training data pairs are readily available. By contrast, active learning is the case where the supervisor can be actively queried to request labels for only a part of the training data. The remaining training labels can be inferred using a ML model. This is particularly useful when a large amount of unlabeled data are available during the training of a supervised model, but labeling data is expensive[176].
- **Batch vs Online :** Batch learning is performed on all available dataset. By contrast, online learning is where the dataset is provided to the learning algorithm one example at a time. Online learning is useful when we have limitations on computation and storage that prevent the possibility to process the whole dataset as a batch.

3.2.2 The Learning Process

The machine learning process is typically composed of the following steps.

Problem Definition

Once the need to use machine learning is established to solve the given problem, several elements must be defined and acquired. As discussed at the beginning of this

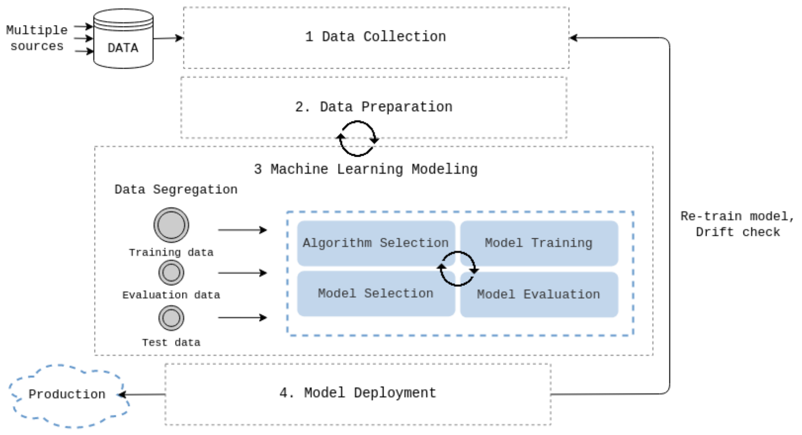


Figure 3.3: An overview on the machine learning workflow.

Chapter 3.2, these elements are: class of tasks, measure of performance to be improved, and source of experience. Class of tasks consists in the space of hypothesis. In supervised learning, the space of hypothesis stands for space of all possible hypotheses for mapping inputs to outputs. Measure of performance consists in a measure for evaluating the machine learning model. The measure that qualifies the project as being successful and that we aim to achieve as objective, should be established from the start. Source of experience consists in observations of the phenomenon we want to model.

Data Collection

The first step of a machine learning workflow is to collect data for analysis. However, *How much data is needed?* The amount of data needed for good performance depends on several factors, such as the complexity of the learning problem, the complexity of the machine learning model that will be used, the target performance to achieve, project constraints (time and computation), etc. It is therefore difficult to accurately estimate the minimum amount of data required for a successful learning project.

Data collection can broadly categorized into two aspects [208]: *data acquisition* and *data labeling*. The goal of data acquisition is to get datasets that can be used for the machine learning process. Data acquisition is generally performed through data discovery, data augmentation and/or data generation. Data discovery is the process of sharing and searching new relevant datasets. In the era of big data and with the growth in open data initiatives, sharing datasets among data holders or releasing open datasets to the public frequently occurs. However, the representativeness of data available and discoverable online with respect to the target ML task often remains a challenge.

Modern machine learning models, namely deep learning, require large amount of data to train, which are often not available. A common solution to mitigate data scarcity is to augment data. Data augmentation consists in acquiring new data in order to augment an existing dataset. Data augmentation can be performed by synthetically altering *already existing data* to create modified copies. For instance, in case of images, common modifications include changing orientation, location, scale or brightness. Otherwise, external datasets may be acquired and integrated to existing data. If no dataset is available for the machine learning task, data are generated. Data generation can be performed manually through crowd-sourcing or automatically. An increasingly popular approach to generate synthetic data automatically is through Generative Adversarial Networks (GANs) [93]. GANs offers a low cost and flexible way to generate plausible new examples from an *existing distribution of examples*. Very briefly, a GAN is composed of two contesting sub-networks, a *generator* and a *discriminator*. The generator is responsible for generating new data examples, while the discriminator is responsible for differentiating synthetic examples from true examples. The generator thus needs to learn how to produce sufficiently realistic examples in order to reliably trick the discriminator.

In case of supervised ML setting, once sufficient amount of data is acquired, examples should be labeled. Data labeling may be performed manually via crowd-sourcing. A time efficient crowd-based approach for labeling is through *active learning*. Active learning is the process of manually labeling just a subset of the available examples and infer the remaining labels automatically using a ML model. A second approach consists in weakly annotating a large amount of data (i.e. *weak labels*). Weak supervision branch argues that weakly labeled data in large amounts can be employed to create more accurate models than when using strongly labeled data alone.

Data Preparation

Data preparation mainly consists in transforming data to a format that is amenable to processing by machine learning algorithms. Gartner[1] defines data preparation as "an iterative and agile process for exploring, combining, cleaning and transforming raw data into curated datasets for data integration, data science, data discovery and analytics/business intelligence use cases". Data cleaning is the process of detecting and removing *noisy* data. Noise is defined as the random error or variance in a measured variable. Noise in data is an effective problem for ML algorithms, as it prevents them from learning properly and causes them to miss out patterns in data.

In the context of ML, as the dimensionality (i.e. number of features) of data increases, the amount of data that is required to deliver a reliable analysis also rises exponentially. Bellman referred to this phenomenon as "*Curse of Dimensionality*" [21]. High-dimensional data are, therefore, challenging as they cause unnecessary computational complexity. The problem with high-dimensional data can be mitigated with *dimensionality reduction*, which consists in projecting the data onto a lower-dimensional subspace, while preserving information as much as possible. Dimensionality reduction techniques are proposed and implemented by using *feature extraction* and *feature selection*. Feature extraction, sometimes referred to as feature transformation, aims to extract a low-dimensional representation of data. Some feature extraction techniques

[1] https://www.gartner.com/reviews/market/data-preparation-tools

are principal component analysis (PCA) [1] and auto-encoders (AE) [23]. Deep auto-encoders are also widely used for feature extraction tasks. Unlike PCA, AE relies on a non-linear transformations. The goal is to map inputs into a lower-dimensional latent space, that enables decoding back to the original input space with minimal information loss. The basic architecture (see Figure 3.4) of an AE is decomposed into two components: an encoder that maps the input into a latent representation, and a decoder that maps the latent representation to a reconstruction of the original input. Different types of AE exist [23], such as denoising AE, variational AE, etc.

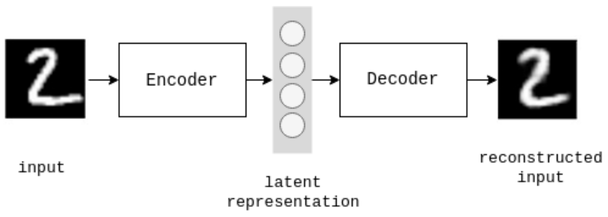


Figure 3.4: Basic architecture of an auto-encoder.

Feature selection simplifies a machine learning problem by choosing which subset of the available features should be used. There are three categories of feature selection methods [135]: filter methods (also called screening or ranking methods), wrapper methods and embedded methods. The simplest approach to feature selection is the filter method. It evaluates the relevance of each feature individually to the learning problem, and then takes the top K features, where K is chosen based on some trade-off between performance and complexity [135].

Machine Learning Modeling

The core of a ML workflow is to design and to train a model that is capable of best explaining some data or phenomena, or to predict future data. Let us consider an input vector $\mathbf{x} \in \mathcal{X}$ to a decision making process, and its corresponding output $y \in \mathcal{Y}$. y may be continuous (regression) or categorical (classification). We suppose that there is a function $f : \mathcal{X} \mapsto \mathcal{Y}$ that describes the approximate relationship between $\mathbf{x}$ and y. The goal of machine learning is therefore to provide an approximation of function f with a model or estimate $\hat{f}$ (often called hypothesis). The learning algorithm chooses $\hat{f}$ from a set of candidate functions, which we call a hypothesis space $\mathcal{H}$. The hypothesis space $\mathcal{H}$ is generally an arbitrary choice when no prior knowledge is available about the true underlying function f. Otherwise, the best hypothesis space $\mathcal{H}$ is selected using the prior knowledge. For instance, $\mathcal{H}$ can be the set of all linear function, from which a linear regression algorithm will choose to fit the data.

An hypothesis $f(.;\theta,\phi) \in \mathcal{H}$ is generally controlled by a set of hyper-parameters ϕ and a set of parameters θ. A hyper-parameter controls the learning algorithm itself. It differs from a parameter in the fact that it can not be fitted to training data during the training process. In that sense, a better notation would be $f_\phi(.;\theta)$. Examples of hyper-parameters are number of clusters k in a k-means model [115] or degree of the polynomial in Polynomial regression [188]. Running a learning algorithm over a training dataset with different hyper-parameter settings will result in different models. In a machine learning context, we typically seek to select the best-performing model from this set. Note that some learning models do not have parameters [92]. Such models are called *non-parametric models* [92], in contrast to *parametric models*.

The learning problem may be decomposed into the following main tasks;

- fitting the model parameters to training data, referred to as *model training* [92],
- estimating the generalization performance of the model on unseen data, referred to as *model evaluation* [203],
- selecting the best-performing model from a given algorithm's hypothesis space by tweaking the learning algorithm , referred to as *hyper-parameters tuning* or *model selection* [97, 203],
- identifying the best-suited learning algorithm for the problem at hand, referred to as *algorithm selection* [203].

For modeling, data are often split into three *distinct* subsets for the purpose of training, validation, and testing the model. Training data are used to adjust parameters θ of the model. Validation data are used to fine-tune hyper-parameters ϕ of the learning algorithm. Test data are used to estimate the generalization performance (or *generalizability*) of the model. Generalization performance of a model refers to its ability to adapt to new, previously unseen examples [92]. Thereby, it represents the true performance of a model when deployed in the real world.

However, holding out data reduces the amount available for training. This can be mitigated by performing *cross-validation* [69]. The idea is as follows : we randomly split the training data into K non-overlapping subsets, called *folds*, hold out each fold $k \in 1...K$ while training on the remaining, test each trained model on the examples of the fold it did not see, average the K results to evaluate how well the particular hyper-parameter setting does [69]. Figure 3.5 illustrates the k-fold cross validation scheme.

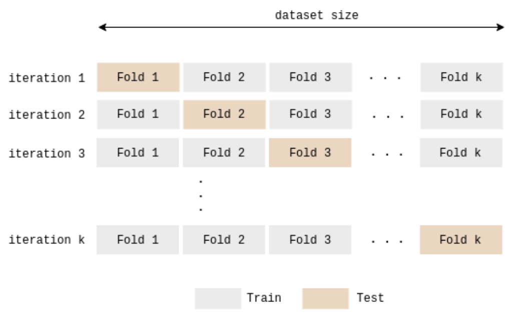


Figure 3.5: Illustration of k-fold cross-validation scheme.

Model Deployment and Performance Monitoring

Building the model is generally not the end of a machine learning workflow. The next step is deployment and integration of this model into an existing production environment. Several challenges arise with model deployment, namely scalability, portability and reproduciblity. Different strategies can be adopted to deploy a pre-trained model. For example, we can wrap the model and serve it as a web-service.

A deployed model also needs to be actively tracked and monitored. This allows to get feedback in case of *concept drift* [256]. Concept drift refers to changes over time in the data distribution. This results in significant deterioration of the model performance. Quantitatively measuring this phenomenon will allow early detection and handling, namely with performance scoring. One way to handle concept drift is by periodically updating the model with more recent data [159]. This prevents completely putting aside a model to train a new one, which is costly. Another way is to build an ensemble of multiple models to boost their performance [159]. Note that concept drift is particularly a challenge in case of static models, meaning the case of *batch learning*. By contrast, in the case of *online learning*, models are updated continuously and on the fly.

3.3 Deep Learning

Deep learning (DL) [92] is a specific kind of machine learning. It refers to a set of machine learning techniques, namely neural networks which learn effective representations of data with multiple levels of abstraction. In contrast to “classic” machine learning, in which representations are manually refined through feature engineering, deep learning aims to incorporate and completely automate this step. This has substantially simplified machine learning workflows, and successfully replaced multi-stage hand-tuned pipelines with *end-to-end* deep learning model [83].

Neural networks (NNs), interchangeably referred to as artificial neural network (ANNs), are complex programs that involve more parameters than a human can tweak. Deep learning does not require human intervention to learn, but instead requires a large amount of relevant data. This concept is referred to by some researchers as *Software 2.0* [120].

A brief glance back into the history of deep learning reveals three major waves [92]: DL known as *Cybernetics* during 1940–1960, DL known as *Connectionism* during 1980–1990, and the current emergence under the name "deep learning" beginning in 2006. The history of NNs [7] began in 1943 when neuro-physiologist Warren McCulloch and mathematician Walter Pitts proposed the first computational model of a neuron by mimicking the functionality of a biological neuron, which was called *threshold logic unit* [171]. The first modern neural network was introduced by Frank Rosenblatt in 1957 under the name *Perceptron* [209]. Minsky and Papert published a book to argue about severe limitations of Perceptrons in 1969 [173]. This has contributed in discrediting and halting research in the field of neural networks for almost a decade, hence the naming "*AI winter*". It was until 1985 that the field was revitalized by the presentation of *back-propagation* by Geoffrey Hinton [4]. Hinton showed that back-propagataion can successfully train deep neural networks (more than two or three layers). It should be mentioned that back-propagation was first introduced by Paul Werbos, in the context of neural networks training. However, this approach was not popularized until 1985. During the ensuing neural network revival in the 1980s, key works had laid the foundation of modern deep learning. However, the introduction of support vector machines with non-linear kernels by Vladimir Vapnik [57] had contributed in stalling neural networks research. The third wave emerged in 2006 with the published work of Hinton [102, 103].

One of the major breakthroughs of deep learning happened in 2012 when Alex Krizhevsky designed Alexnet [133], a convolutional neural network that outperformed the then current state-of-the-art in the ImageNet competition, and brought error rate down to 15.3%, compared to 26.2% achieved by second best team. Even though the main idea of training deep neural networks was not new, these results had successfully demonstrated the capability of deep learning to solve a complex task on a large and complicated dataset like ImageNet. This is attributed in great part to the increase of computational power and memory capacity of machines. Deep learning grew to prominence and had attracted great interest since 2012. Nowadays, deep learning is delivering state-of-the-art results in many fields.

3.3.1 Feed-forward Networks

Feed-forward Neural Networks (FNNs) [92], often called Multi-Layer Perceptrons (MLPs), are the most basic and widely used artificial neural networks. Hence, it is sometimes referred to as "*vanilla*" neural networks.

FNNs [92] consist on connected neurons in a finite directed acyclic graph. Given the absence of cycles, a FNN can be arranged into different layers in which each layer can only communicate with the upper layer. These models are called *feed-forward* because information flows unidirectionally from input layer to output layer. As a whole, a FNN is a chain of composed functions, where each layer is a function that takes as input the output of the previous layer. This chain of composed functions is optimized to perform a task. For instance, let's suppose that a network is composed of three functions $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$. The final function f is written as $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$ [92]. The *input layer* corresponds to the input vector $\mathbf{x}$, $f^{(1)}$ is called the *first layer*, $f^{(2)}$ is called the *second layer*, and so on. The output layer corresponds to predictions $f(\mathbf{x})$. Layers that are situated between the input layer and the output layer are also called *hidden layers* given that their outputs are not directly observable from the training data. The length of the chain is the *depth* of the network. Deep learning is defined in terms of the depth of the neural networks architecture. However, there is no universally agreed upon threshold of depth that divide "shallow" neural networks and "deep" neural networks.

Figure 3.6 shows an example of a FNN with a single hidden layer. Input layer is composed of D neurons, corresponding to each of the D input variable $x_1, x_2, ..., x_D$. Hidden layer is composed of M neurons. Output layer is composed of K neurons. Each j^{th} neuron of the hidden layer is obtained by :

- Linear combination a_j of input vector $\mathbf{x} = (x_i)_{i=1...D}$, written as:

$$a_j^{(1)} = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + b_j^{(1)} \forall j \in \{1...M\} \tag{3.1}$$

  D is the total number of input variables. The superscript (1) indicates that the corresponding parameters are in the first layer of the network. Parameters w_{ji} are referred to as *weights*, whereas parameters b_j are referred to as *biases*. Quantities a_j are known as *activations*.

- and a nonlinear transformation $f : \mathbb{R} \mapsto \mathbb{R}$ as follows :

$$h_j^{(1)} = f\left(a_j^{(1)}\right) \forall j \in \{1..M\} \tag{3.2}$$

  $h_j^{(1)}$ are referred to as *hidden units*, f is referred to as *activation function*.

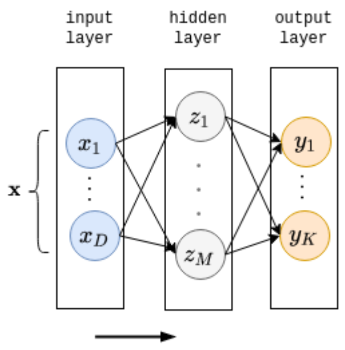


Figure 3.6: Architecture of a Deep Feed-Forward Neural Network; The input layer is in blue, the hidden layers are in gray and the output layer is in orange. The arrows represent the forward pass across different layers.

Output vector $\hat{y}$ is calculated in a same manner. *Output unit activations* are obtained using a weighted linear combination of hidden units, followed by an element-wise transformation using a nonlinear activation function, as follows:

$$\begin{aligned} a_k &= \sum_{j=1}^{M} w_{kj}^{(2)} f(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + b_j^{(1)}) + b_k^{(2)} \forall k \in \{1...K\} \\ \hat{y}_k &= f\left(a_k\right) \end{aligned} \tag{3.3}$$

We can combine these various stages to give the overall network function in Figure 3.6 that takes the form :

$$\hat{y}_k(\mathbf{x}, \mathbf{w}) = f\Big(\sum_{j=1}^{M} w_{kj}^{(2)} f\Big(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + b_j^{(1)}\Big) + b_k^{(2)}\Big) \forall \mathbf{x} \forall k \in \{1...K\} \tag{3.4}$$

where the set of all adjustable parameters (weights and biases) were grouped together into a vector **w**. Different terminology is used in literature for counting the number of layers composing a neural network. Some studies count the number of layers of units, including input and output layers. Others count the number of hidden layers. Following the first mentioned terminology, the network shown in Figure 3.6 is described as three-layer network, whereas, when following the second terminology, it is described as one-layer network. In this manuscript, we follow the terminology that describes networks by the number of layers of adjustable parameters. Figure 3.6 is therefore called a two-layer network.

Many choices of activation functions can be made in the framework of NNs, namely sigmoid (also called logistic) function, hyperbolic tangent (tanh). They respectively calculate $\sigma(x) = \frac{1}{1+e^{-x}}$ and $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Rectified Linear Unit (ReLU) is also widely used for deep learning, since it substantially reduces the computational cost for training of the network. ReLU calculates $f(x) = \max(0, x)$.

The activation function at the output layer depends upon the task. For regression, we generally use a linear function, $f(x) = ax$.

NNs as universal approximators Neural networks are known to be *universal approximators*. Theoretical results show that, a two-layer network with no-polynomial activation function can approximate any continuous function on a compact set of $\mathbb{R}^n$ to any desired degree of accuracy [61, 108, 109]. Although such result proves the existence of a solution, no methods are provided for finding it. Furthermore, the main challenge is how to find the most suitable parameters given a set of training data. Motivated by the current success of deep learning, several studies investigated the approximation capabilities of deep neural network, compared to 'shallow' neural networks. An interesting result [64] shows that there exist a deep sum-product network that can not be approximated by shallow sum-product networks, unless they use an exponentially larger number of neurons.

Despite their power, FNN present several limitations. Notably, they rely on the assumption of independence between training examples [155]. However, this is not the case when data point are related in time or space, i.e. sequence of video frames or audio signal.

3.3.2 Error Back-propagation

Neural networks accept an input **x** and outputs a prediction $\hat{y}$. Information flows from input units, through hidden units in each layer, and to the output units. This is called *forward propagation* or *forward pass*. During training, forward propagation results in an error. *Back-propagation* [215] propagates the error from output units back towards the input units, in order to compute gradients, and make updates accordingly to all parameters in all layers. The process of forward propagation and back-propagation is repeated until reaching an optimal error. As such, back-propagation is a method for computing gradients, which are used by training algorithms like stochastic gradient descent (SGD).

Typically, error functions in neural networks are non-convex, which means that finding “provably” optimal solution is NP-hard. Moreover, a very large number (possibly infinite) of local minima is present in deep networks error functions. However, these local minima have almost equivalent cost for sufficiently deep and/or large neural networks. Reaching a local minimum is therefore sufficient [91, 53].

3.4 Time Series Forecasting

A time series is a set of observation ordered sequentially in time, observed at a discrete set of evenly spaced time intervals; $\{x_t\}_{t=1..N}$, where N is the length of the time series.

Usually, time series analysis can be divided into univariate and multivariate analysis. Univariate time series refers to a time series containing a single observation recorded sequentially over time, such as daily energy consumption of a building. Multivariate time series is used when a group of time series variables are involved and their interactions are to be considered. Each variable thus depends not only on its past values but also has some dependency on other variables. For instance, energy consumption in buildings is affected by several time-variant exogenous factors like weather conditions, human activities, holidays, etc.

The goal of time series forecasting is to predict future values of a target $y_{i,t}$ for a given entity i at a given time t. An entity consists in a logical collection of temporal information, such as vital signs of different patients in healthcare, stock prices of different companies in finances, or power consumption measurements of different buildings in the field of energy efficiency. Time series forecasting models can be written as:

$$\hat{y}_{i,t+1} = f(y_{i,t-k:t}, \mathbf{x}_{i,t-k:t}) + e(t) \tag{3.5}$$

where $\hat{y}_{i,t+1}$ is the one-step ahead model prediction at time $t+1$, $y_{i,t-k:t}$ and $\mathbf{x}_{i,t-k:t}$ are previous values of respectively the target and exogenous variables over a look-back time window of size k, $e(t)$ is the noise term and $f(.)$ is the actual prediction model [153].

3.4.1 Multi-step Time Series Forecasting

Multi-step ahead time series forecasting consists in predicting h next values of a target $\{y_{i,n+1}, y_{i,n+2}, ..., y_{i,n+h}\}$ from past n observations of the target $\{y_{i,1}, y_{i,2}, ..., y_{i,n}\}$ (and some exogenous variables if applicable), $h > 1$ is the forecasting horizon, n is the number of observations of the historical time series. There are mainly three strategies for multi-step time series forecasting [240, 230]: (1) recursive strategy, (2) direct strategy, and (3) multi-input multi-output (MIMO) strategy. Hybrid strategies that combine two or more strategies also exist, such as DirRec (direct and recursive) and DIRMO (direct and MIMO).

In what follows, let $\{y_{i,t:t+l}$ be a shorthand notation for $\{y_{i,t}, y_{i,t+1}, ..., y_{i,t+l}\}$.

Recursive strategy (also called iterative or multi-stage) [230] is the most intuitive strategy. It consists in recursively iterating a single step model to enable multi-step forecasting. This implies that previous predictions are subsequently introduced with the true observations of the target as inputs to predict future time steps. Figure 3.7 illustrates the recursive strategy. For a prediction horizon H, a trained single step model $\hat{f}$, multi-step prediction is performed as follows:

$$\hat{y}_{i,t+h} = \begin{cases} \hat{f}(y_{i,t-k+1:t}) & \text{if } h = 1 \\ \hat{f}(\hat{y}_{i,t+1:t+h-1}, y_{i,t-k+h:t}) & \text{if } 2 \leq h \leq k \\ \hat{f}(\hat{y}_{i,t+1:+h-k}) & \text{if } k < h \leq H \end{cases} \tag{3.6}$$

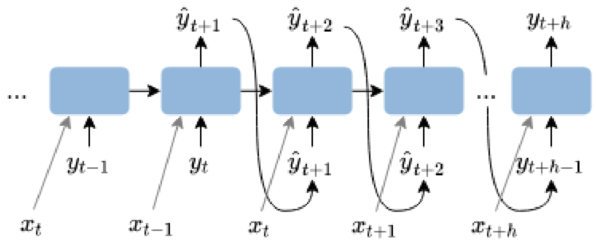


Figure 3.7: Recursive strategy.

Given that a potential error is induced at each one-step ahead forecast, re-using of predictions will create a feedback loop, and will therefore lead to accumulation of errors over the forecasting horizon.

Direct strategy (also called independent) [230] consists in predicting each future time step independently. This implies that h prediction models are trained. Let $\hat{f}_h$ be the trained one-step model for predicting time step h, such as:

$$\hat{y}_{i,t+h}\hat{y}_{i,t+h} = \hat{f}_h(y_{i,t-k+1:t}) \quad \text{for } 1 \leq h \leq H \tag{3.7}$$

In this strategy, there is no accumulation of errors because only true observations of target variable are used for the prediction at future time steps. However, the main issue of the direct strategy is that underlying dependencies between predictions are not taken into consideration. Given that one model is trained at each time step, computational and maintenance cost is also significant, especially as the forecast horizon increases.

DirRec strategy DirRect is a hybrid strategy that combines both RECursive and DIRect strategies [230]. It uses a different model at each future time step (direct strategy) and feeds predictions from previous time steps into the inputs (recursive strategy). DirRec strategy can be written as:

$$\hat{y}_{i,t+h} = \begin{cases} \hat{f}_h(y_{i,t-k+1:t}) & \text{if } h = 1 \\ \hat{f}_h(\hat{y}_{i,t+1:t+h-1}, y_{i,t-k+1:t}) & \text{if } 2 \leq h \leq H \end{cases} \tag{3.8}$$

DirRec strategy avoids the accumulation of errors problem encountered in the recursive strategy, and the conditional independence assumption between predictions across the horizon encountered in the direct strategy. However, by definition, the input set in DirRec strategy grows linearly with horizon H, thus yielding significant computational cost.

MIMO strategy Recursive, direct and DirRec strategies rely on multi-input and single-output models [240]. By contrast, MIMO strategy considers a multi-input and multi-output model. This is motivated by the need to model stochastic dependencies between future values of the time series and past observations [32]. MIMO strategy can be written as:

$$\hat{y}_{i,t+1:t+h} = \hat{f}(y_{i,t-k+1:t}) \tag{3.9}$$

MIMO strategy alleviates the conditional independence assumption between H predictions of the direct strategy. It also avoids accumulation of errors as in the recursive strategy. However, MIMO strategy uses only one model to predict all the horizons. This may constrain the flexibility of the forecasting approach, and hence may induce a highly biased model [240].

DIRMO strategy (also called Multi-Input Several Multi-Output (MISMO)) DIRMO combines the DIRect and the MIMO strategies [240]. DIRMO forecasts the horizon in portions, where for each portion, there is a MIMO predictor. For a prediction horizon H, the prediction task is decomposed to $n = \frac{H}{s}$ prediction sub-tasks, where $s \in \{1...H\}$ is the output size of each sub-task. DIRMO strategy can be written as:

$$\hat{y}_{i,t+(p-1)s+1:t+ps} = \hat{f}_p(y_{i,t-k+1:t}) \tag{3.10}$$

where $p \in \{1...n\}$. Notice that for $s = 1$, the approach is equivalent to the direct strategy. Whereas, for $s = H$, the approach is equivalent to the MIMO strategy.

3.4.2 Overview on Machine Learning Techniques

Traditionally, time series forecasting mostly relies on statistical models like vector auto-regression (VAR) [157], and auto-regressive integrated moving average (ARIMA) [184] and its many variations. ARIMA was first proposed by Box and Jenkins [38]. ARIMA model has elements of other time series models including autoregressive (AR) model and moving average (MA) model. The *multivariate* version of ARIMA is called ARIMA with eXogenous inputs (ARIMAX). However, despite their popularity, ARIMA and its variants are generally not used for high-dimensional multivariate time series forecasting tasks due to their high computational demand [140]. Another main issue with ARIMA is the linearity assumption on the data. If this assumption is violated, the statistical model may be biased and misleading as it fails to capture non-linear data relationships.

VAR is the extension of AR models to *multivariate* problems. VAR has proven to be powerful technique due to its flexibility and simplicity. It is also successfully used for *multi-step* ahead time forecasting. Nonetheless, like ARIMA, VAR fails to capture complex non-linear relationships of multivariate time series data. VAR algorithm is also limited by linear complexity over the width of the temporal window and quadratic complexity over the number of variables [140]. VAR is thus prone to over-fitting when modeling long-run temporal patterns. This also hinders the performance for highly dimensional multivariate time series.

Gaussian process is a way of performing a non-parametric Bayesian inference. GPs provide a Bayesian prior distribution over unknown functions, and then sequentially update this distribution by conditioning on the data. GPs excel at capturing complex dynamics and nonlinear relationships in data. The main shortcoming for GP is the high computational complexity $\mathcal{O}(n^3)$ for a number of observations n. This hinders GP from scaling up to large datasets.

Time series forecasting may be seen as a standard regression setting. Common supervised regression models can therefore be applied, such as support vector machine (SVM). A major breakthrough in time series forecasting area occured with the introduction of SVM [57]. Support vector machine was applied to regression, and was therefore called support vector regression (SVR). SVR is practically simple and SVR techniques employ kernels in order to capture non-linear patterns in time series data. Commonly used kernels are the polynomial kernel and the radial basis function kernel. Extension of SVR to multi-step ahead setting were proposed in [167, 221, 18].

Over the past decade, deep learning has achieved remarkable success in a broad range of applications, namely time series forecasting. Deep learning [92], as subset of ML based on deep ANN, attempts to extract meaningful representations with different levels of abstraction using nonlinear transformations [23]. Neural networks alleviate the need for manual feature extraction. For multi-step ahead forecasting with multivariate time series, several deep learning techniques were proposed [78], namely convolutional neural networks (CNN) and recurrent neural networks (RNN).

Recurrent neural networks (RNN) [156] have been historically proposed for sequence modeling tasks. It was therefore naturally applied to time series forecasting. The reason behind the effectiveness of RNN comes from its ability to capture past information from data, and use it to inform upcoming sequence steps. Two variants of RNN in particular, namely the Long Short Term Memory (LSTM) [284, 41] and the Gated Recurrent Unit (GRU) have significant success in several sequential data modeling applications. LSTM and GRU offer the ability to capture long-term dependencies, while addressing vanishing/exploding gradients problem [192] often encountered in recurrent networks. This is essentially achieved via special gates that determine which past information should be passed and which should be forgotten [89].

Convolutional neural networks (CNN) [142] have gained a lot of attention with their applications in the fields of computer vision and pattern recognition. Given their success, CNN were adopted to time series forecasting tasks. CNNs have powerful feature extraction capabilities. Furthermore, Dilated convolutions allow to capture long-term temporal dependencies. To enable multi-step ahead forecasting, a CNN is usually combined with a RNN to form a hybrid model.

3.4.3 Support Vector Regression

Support vector machine (SVM) [57] is a linear model that is proposed for classification and regression problems. Support vector regression [228] have been successfully and widely employed for time series regression in the context of building energy modeling. The main idea behind SVM is to find a hyperplane that linearly separates the data with the largest *margin*. With support vector regression (SVR), the goal is also to find a hyperplane that can be used for regression.

Define a dataset $D = \{(\mathbf{x}^{(i)}, y^{(i)}) \subset \mathcal{X} \times \mathcal{Y}, \text{where} \quad i = 1...N\}$ where N is the size of the dataset, $\mathbf{x}^{(i)} \in \mathbb{R}^n$ is the input feature vector and $y^{(i)} \in \mathbb{R}$ is its corresponding label. In case of multivariate time series forecasting, $\mathbf{x}$ is the vector of past values of the target y combined with the exogenous features. Otherwise, it will be simply the vector of past values of the target. For a given tolerance ϵ, the ϵ-insensitive SVR consists in finding a function $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$, such that all data points are within a strip bounded by two parallel hyperplanes (as shown in Figure 3.8). All data points are therefore allowed to deviate at most a certain margin $\epsilon \geq 0$ from the targets to be predicted.

Mathematically, this is achieved through the following optimization problem [228].

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \|\mathbf{w}\|^2 \\ \text{s.t.} \quad & \begin{cases} y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)} - b \leq \epsilon \\ \mathbf{w}^T\mathbf{x}^{(i)} + b - y^{(i)} \leq \epsilon \end{cases} \end{aligned} \tag{3.11}$$

Depending on the value of ϵ, it is possible that a function f, that satisfies these constraints, does not exist. A relaxation extension is to introduce slack variable $\xi_i \geq 0$

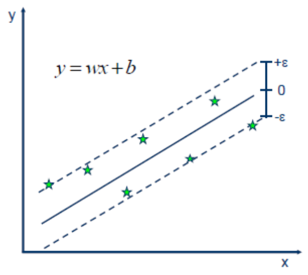


Figure 3.8: Illustration of a support vector regression function [223].

(for the upper boundary) and $\xi_i^* \geq 0$ (for the lower boundary) for each point (as shown in Figure 3.9. The slack variables allow regression errors. The optimization problem is therefore modified as follows [228].

$$\begin{aligned} \min_{w,b,\xi,\xi^*} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\ \text{s.t.} \quad & \begin{cases} y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)} - b \leq \xi_i + \epsilon \\ \mathbf{w}^T\mathbf{x}^{(i)} + b - y^{(i)} \leq \xi_i^* + \epsilon \\ \xi_i \geq 0, \xi_i^* \geq 0 & \forall i = 1...N \end{cases} \end{aligned} \tag{3.12}$$

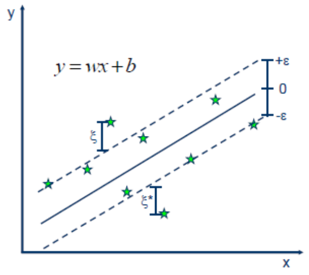


Figure 3.9: Illustration of a support vector regression function [223].

Here ξ and ξ^* are shorthand notations for respectively $\xi = \{\xi_i\}_{i=1}^N$ and $\xi^* = \{\xi_i^*\}_{i=1}^N$. The regularization constant C determines the trade-off between the achieving low training errors and the overfitting. The bigger C is, the more errors are accepted. The regression problem may be solved using Lagrangian multipliers $\alpha_i, \alpha_i^*, \eta_i, \eta_i^*$, to incorporate the constraints as follows :

$$
\begin{aligned}
L &= \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) - \sum_{i=1}^{N}(\eta_i\xi_i + \eta_i^*\xi_i^*) \\
&- \sum_{i=1}^{N}\alpha_i(\xi_i + \epsilon - y^{(i)} + \mathbf{w}^T\mathbf{x}^{(i)} + b) \\
&- \sum_{i=1}^{N}\alpha_i^*(\xi_i^* + \epsilon - \mathbf{w}^T\mathbf{x}^{(i)} - b + y^{(i)})
\end{aligned}
\tag{3.13}
$$

Optimality can be achieved by solving the equations $\frac{\partial L}{\partial \mathbf{w}} = 0$, $\frac{\partial L}{\partial b} = 0$, $\frac{\partial L}{\partial \xi} = 0$, and $\frac{\partial L}{\partial \xi^*} = 0$. Namely, **w** can be written as a linear combination of the training observations as follows [228].

$$
\mathbf{w} = \sum_{i=1}^{N}(\alpha_i + \alpha_i^*)\mathbf{x}^{(i)} \tag{3.14}
$$

In addition, we have

$$
\frac{\partial L}{\partial b} = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i) = 0 \tag{3.15}
$$

And

$$
\begin{aligned}
\frac{\partial L}{\partial \xi_i} &= C - \eta_i - \alpha_i = 0 \quad &\Rightarrow \eta_i = C - \alpha_i \\
\frac{\partial L}{\partial \xi_i^*} &= C - \eta_i^* - \alpha_i^* = 0 \quad &\Rightarrow \eta_i^* = C - \alpha_i^*
\end{aligned}
\tag{3.16}
$$

Given that $\eta_i \geq 0$ and $\eta_i^* \geq 0$, we have $\alpha_i \in [0, C]$ and $\alpha_i^* \in [0, C]$. For chosen α and α^*, we compute :

$$
b = y^{(k)} - \sum_{i=1}^{N}(\alpha_i + \alpha_i^*)\mathbf{x}^{(i)T}\mathbf{x}^{(k)} \tag{3.17}
$$

Therefore, for a test point $\mathbf{x} \in \mathbb{R}^n$, the predicted value y is defined as

$$
y = \sum_{i=1}^{N}(\alpha_i + \alpha_i^*)\mathbf{x}^{(i)T}\mathbf{x} + b \tag{3.18}
$$

Non-linear case One of the main forces of SVM is that it can be easily extended to linearly inseparable data [34]. This is done by mapping the data in the input space into a higher-dimensional space where they become linearly separable (as shown in Figure 3.9). This is known as kernel method. One problem is that the input vectors may be mapped to a very high-dimensional space, and possibly infinite, which results in very expensive computations of dot products. An easy solution here is the use of *kernel trick*. Kernel trick allows to operate in the input space without the need to compute new coordinates in the transformed feature space, via replacing the dot product

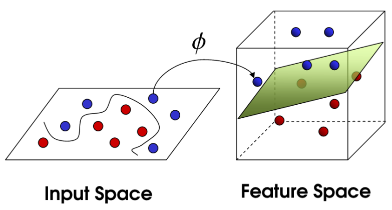


Figure 3.10: Illustration of kernel mapping in SVM for binary classification task [177]. ϕ maps input data point to a higher dimensional feature space. Transformed representations of data points are linearly separable in the new feature space.

$< \phi(\mathbf{x}), \phi(\mathbf{z}) >$ with a function $K(x, \mathbf{z})$. As a result, we will only use K in the training algorithm, without needing to explicitly compute or even know what the feature mapping ϕ is.

Notice that input vectors only appear in the form of dot products in the training problem, i.e. equation 3.18. Then we just replace dot products with the kernel as follows.

$$y = \sum_{i=1}^{N} (\alpha_i + \alpha_i^*) K(\mathbf{x}^{(i)}, \mathbf{x}) + b \tag{3.19}$$

In addition to support vector machines, kernels are used in other algorithms [106], such as non-linear variant of PCA [211].

For a function to be a valid kernel, necessary and sufficient condition is as follows: Let G be a kernel matrix or Gram matrix be a $N \times N$ matrix where each entry i, j corresponds to $G_{i,j} = K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ where $\{\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\}$ are feature vectors of the dataset $\mathcal{D}$. $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is a valid kernel if and only if matrix G is symmetric, positive semi-definite. This is known as the Mercer's theorem.

Several kernels are commonly used, namely :

- **Linear :** $K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$,
- **Polynomial :** $K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^T \mathbf{z})^p$ where $p \in \mathbb{N}$ is the degree of the polynomial,
- **Radial basis function (RBF)**, also known as Gaussian or radial :
  $K(\mathbf{x}, \mathbf{z}) = \exp(-\gamma \|\mathbf{x} - \mathbf{z}\|^2)$ where γ is a free parameter.

Multi-output SVR So far, the explained SVR algorithm can be applied to multivariate one-step ahead time series forecasting. However, it would need to be adopted for multi-step ahead time series forecasting. MIMO SVR (M-SVR) [167, 221, 18] was proposed. Compared to the standard SVR, output y becomes multi-dimensional $\in \mathbb{R}^M$. The objective function therefore becomes:

$$\begin{aligned} \min_{\mathbf{W},\mathbf{b},\xi,\xi^*} \quad & \frac{1}{2}\sum_{j=1}^{M}\|\mathbf{w}^{(j)}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\ \text{s.t.} \quad & \begin{cases} y^{(i)} - \mathbf{w}^{(j)T}\mathbf{x}^{(i)} - b^{(j)} \leq \xi_i + \epsilon & \text{for } j = 1...M \\ \mathbf{w}^{(j)T}\mathbf{x}^{(i)} + b^{(j)} - y^{(i)} \leq \xi_i^* + \epsilon & \text{for } j = 1...M \\ \xi_i \geq 0, \xi_i^* \geq 0 & \forall i = 1...N \end{cases} \end{aligned} \tag{3.20}$$

where $\mathbf{W} = \{\mathbf{w}^{(1)}, ..., \mathbf{w}^{(M)}\}$ and $\mathbf{b} = \{\mathbf{b}^{(1)}, ..., \mathbf{b}^{(M)}\}$ are respectively weights and biases.

Another possibility for implementing a multi-output consists in using multiple SVR models as proposed in [280]. These models are trained individually using the same input samples but different targets. As such, for a prediction horizon H, H single-output SVR models are trained, where the h-the model predicts the h-step ahead value. Generated predictions are finally combined and served as a multi-output prediction.

3.4.4 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) [156] are a powerful class of supervised machine learning models that are capable of modeling sequential data. In contrast to a MLP that maps one fixed-sized input to one fixed-sized output, a RNN can handle sequential data of arbitrary sequence length in the input, the output, or more generally both. Figure 3.11 illustrates different types of sequential data that can be handled by RNNs; (1) many-to-many and synced many-to-many networks used for tasks like machine translation by taking a sentence (sequence of words) as input and producing a translated sentence in another language as output, (2) many to one network used for example for sentiment classification by taking a text as input and predicting its sentiment, (3) one-to-many network that can be used for music generation by taking a single musical note as input and producing a piece of music as output.

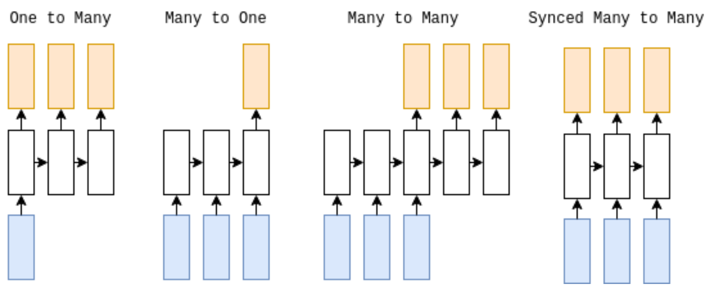


Figure 3.11: Different types of sequential data handled by RNNs [121]; Each rectangle represents a vector while arrows represent operations. The inputs are in blue, the outputs in orange and the hidden states in white. From right to left, we have: (1) sequential output (2) sequential input (3) sequential input and output (4) Synced sequential input and output..

RNNs rely on the standard computational graph of feed-forward networks. RNNs add the possibility to pass information between adjacent time-steps, and hence the ability to learn long-term dependencies. This is done by sharing parameters across multiple time-steps. Figure 3.12 depicts a simple architecture of a recurrent neural network with one hidden layer.

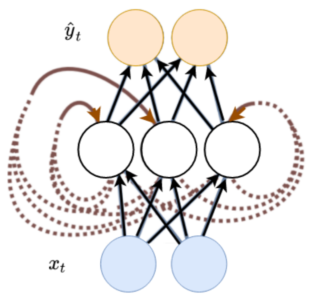


Figure 3.12: A simple one-layer recurrent neural network with one hidden layer; a feed-forward neural network that has a layer for each timestep. At time step t, nodes (white circles) with recurrent edges receive input from the current data point $x^{(t)}$ (blue circles) and from the hidden node values $h^{(t-1)}$ in the network's previous state and calculate the current hidden state $h^{(t)}$. The output prediction $\hat{y}^{(t)}$ (orange circles) is then provided by the output layer.

The dynamics of recurrent networks across multiple time-steps can be seen by "unfolding" it as in figure 3.13. At time step t, hidden nodes with recurrent edges take input vector $x^{(t)}$ and the previous hidden state $h^{(t-1)}$ to compute the current hidden state $h^{(t)}$. The hidden vector $h^{(t)}$ and the output vector $\hat{y}^{(t)}$ are written as.

$$\begin{aligned} h^{(t)} &= \begin{cases} f_h(W_{xh}x^{(t)} + W_{hh}h^{(t-1)} + b_h) & \text{if } t \geq 1 \\ 0 & \text{otherwise} \end{cases} \\ \hat{y}^{(t)} &= f_y(W_{hy}h^{(t)} + b_y) \end{aligned} \tag{3.21}$$

where f_h is the non-linear activation function at the hidden layer, W_{xh} is the weights matrix between the input and the hidden layer, W_{hh} is the weight matrix between the hidden layer and itself at adjacent time, b_h is the bias parameter at the hidden layer which allow each node to learn an offset. The initial hidden state $h^{(0)}$ is often fixed to a vector of zero values, set as a learnable parameter, or initialized based on some other information. f_y is the non-linear activation function at the output layer, W_{hy} is the weight matrix between the hidden layer and the output layer, b_y is the bias parameter at the output layer.

Training a recurrent neural network involves iteratively adjusting the weights and biases, generally using some method of gradient descent and back-propagation through time (BPTT). The most widely used back-propagation algorithm for recurrent networks is back-propagation through time algorithm (BPTT) [257]. Truncated BPTT (TBPTT) [259] is often used as an approximation of BPTT that has the computational efficiency

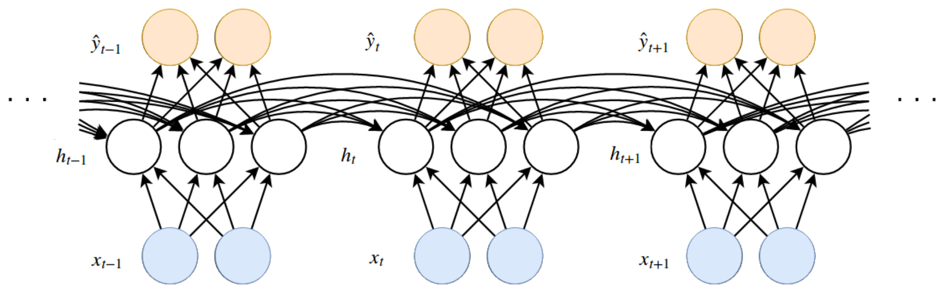


Figure 3.13: Unfolded view of a simple recurrent Neural Network; a feed-forward neural network that has a hidden layer for each timestep. Weights and biases are shared across time steps.

of BPTT, all while tackling its high computational and memory cost. The main idea is to split long sequences into small batches and treating them as separate training cases [237].

A major challenge with training recurrent networks is the vanishing/exploding gradient [191]; The gradient expresses the change in all weights with regard to the change in error. If the gradient is unknown, no further adjustments can be applied to the weights in a direction that will decrease error, and thus, the network ceases to learn. There are two factors that affect the magnitude of gradients; the weights and the non-linear activation functions that the gradient passes through. If either of these factors is smaller than 1, by repeatedly multiplying gradients through every layer during back-propagation, the gradients may vanish fast in time Likewise, if larger than 1, the gradients will decrease exponentially until vanishing. Note that the problem of vanishing/exploding gradient can occur in deep learning in general. However, it is particularly a challenge in RNNs due when learning long-time dependencies. "Unfolding" the RNN will hence result in propagating the gradients through many time steps.

Exploding gradients can be addressed by *gradient clipping* technique, which "clips" gradients whenever their values are greater than a pre-defined threshold. Vanishing gradients may be partially resolved when using TBTT given that it restricts the number of steps over which back-propagation is run. However, this is usually not enough. One of the most commonly used solutions to the vanishing/exploding gradients problem is Long Short-Term Memory (LSTM) architecture.

Long-Short Term Memory LSTM was first proposed in [104] as a solution to the vanishing/exploding gradient problem. The main idea of LSTM is to allow incoming information to alter the state of the memory cell or block it via special gates.

As any classic RNN architecture, there are input layer, recurrent layers and output layer. The hidden layer is constructed by hidden neuron, which is called memory block. A memory block can contain one or several memory cells [104]. A common

LSTM structure is usually composed of four main elements: input gate, self-recurrent connection, forget gate and output gate. The input gate controls the flow of information to the the memory cell. The forget gate aims to determine which past information should be passed and which should be forgotten between memory cells from the prior to the current time step. The output gate controls the flow of information from the memory cell to the next layer.

The common LSTM architecture is defined in terms of the following equations:

$$i^{(t)} = \psi(W_{xi}x^{(t)} + W_{hi}h^{(t-1)} + b_i) \tag{3.22}$$

$$f^{(t)} = \psi((W_{xf}x^{(t)} + W_{hf}h^{(t-1)} + b_f) \tag{3.23}$$

$$u^{(t)} = \phi(W_{xu}x^{(t)} + W_{hu}h^{(t-1)} + b_u) \tag{3.24}$$

$$c^{(t)} = i^{(t)} \odot u^{(t)} + f^{(t)} \odot c^{(t-1)} \tag{3.25}$$

$$o^{(t)} = \psi(W_{xo}x^{(t)} + W_{ho}h^{(t-1)} + b_o) \tag{3.26}$$

$$h^{(t)} = o^{(t)} \odot \phi(c^{(t)}) \tag{3.27}$$

ψ and ϕ are non-linear activation functions. Equation 3.24 is the update operation which is basically the same as standard RNN. Equations 3.22 and 3.26 are respectively the input gate and output gate of the LSTM cell. The main idea of gates is to either allow incoming information to alter the state of the memory cell or block it. All of the gates perform an linear transform followed by a non-linear activation function.

The output of the input gate is later used to get the current cell state, by performing a component wise multiplication respectively with the update operation's output $u^{(t)}$ and the cell state $c^{(t-1)}$ of the previous time-step, as shown in Equation 3.25. We should note that the latter equation is the one that prevents the vanishing/exploding gradient problem from occurring by ensuring that $\frac{\partial c^{(t)}}{\partial c^{(t-1)}} = 1$.

Finally, Equation 3.27 calculates the hidden state, by performing a component wise multiplication between the output of the output gate and the scaled value of the cell state. This hidden output is the one that will be used to compute the predictions $\hat{y}^{(t)}$.

Gated Recurrent Unit Gated Recurrent Unit (GRU) [54] is a simplified version of LSTMs. Similarly to LSTMs, GRU is a variation of RNN architectures that were explicitly designed to deal with vanishing/exploding gradients and efficiently learn long-range dependencies. The dynamics of a gated recurrent unit is defined by the following equations.

$$r^{(t)} = \psi(W_{xr}x^{(t)} + W_{hr}h^{(t-1)} + b_r) \tag{3.28}$$

$$z^{(t)} = \psi((W_{xz}x^{(t)} + W_{hz}h^{(t-1)} + b_z) \tag{3.29}$$

$$\widetilde{h}^{(t)} = \phi(W_{xh}x^{(t)} + W_{hh}(r^{(t)} \odot h^{(t-1)}) + b_h) \tag{3.30}$$

$$h^{(t)} = (1 - z^{(t)})h^{(t-1)} + z^{(t)}\widetilde{h}^{(t)} \tag{3.31}$$

Where $z^{(t)}$ and $r^{(t)}$ are respectively the update gate and the reset gate. $\widetilde{h}^{(t)}$ is a candidate hidden state which is a classic RNN update except of the part where it interpolates a modulation using the reset gate. The final update hidden state interpolates between the candidate state and the previous hidden state in terms of the update gate. Compared to LSTM, GRU has fewer parameters. Therefore, GRU was deployed to reduce computational cost and time.

Sequence to Sequence Model (Seq2Seq) [238] (sometimes referred to as encoder-decoders or auto-encoders) Seq2Seq models are a special architecture of RNNs. They are widely used in a many tasks such as machine translation, image/video captioning and question answering. The main idea is to use two sub-networks; and encoder and a decoder. An encoder takes an input sequence and summarizes information into a *encoder state*. An encoder state aims to encapsulate information that are useful for the target task, and serves as "context" of the decoder. A decoder predict the target sequence given the encoder state. Figure 3.14 illustrates the dynamics of Seq2Seq model.

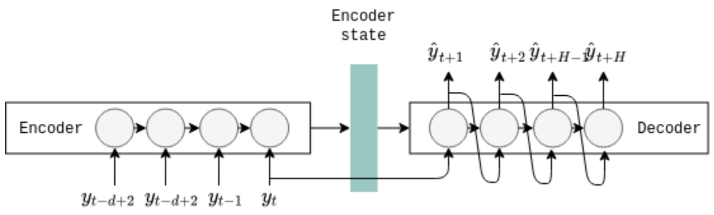


Figure 3.14: The architecture of a sequence-to-sequence model.

Seq2Seq models are usually trained using *teacher forcing* approach [258]. Instead of re-injecting the decoder's predictions into the decoder, teacher forcing uses ground truth target values as the next input for the decoder. This allows faster convergence and more model stability. Teacher forcing updates can be written as:

$$\hat{y}_{t+h} = \begin{cases} \hat{f}(y_t, \mathbf{c}) & \text{if } h = 1 \\ \hat{f}(y_{t+h-1}, h_{t+h-1}) & \text{if } 1 < h \leq H \end{cases} \tag{3.32}$$

where $\hat{y}_t$ is the predicted target at timestep t, y_{t-1} is actual target, h_{t-1} is decoder last hidden state. $\mathbf{c}$ is the encoder context. H is the forecast horizon. $\hat{f}$ is the estimated decoder. Teacher forcing approach is further illustrated in Figure 3.15.

During inference, the ground truth target values are not available, and therefore replaced by the predicted values (as depicted in Figure 3.14. This approach is sometimes referred to as *free running* model or *self-generated* samples. It can be written as:

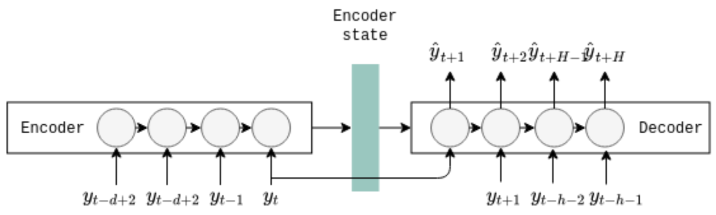


Figure 3.15: The architecture of a sequence-to-sequence model with "teacher forcing" training approach.

$$\hat{y}_{t+h} = \begin{cases} \hat{f}(y_t, \mathbf{c}) & \text{if } h = 1 \\ \hat{f}(\hat{y}_{t+h-1}, h_{t+h-1}) & \text{if } 1 < h \leq H \end{cases} \tag{3.33}$$

Attention mechanisms Encoder-decoder models in general, and Seq2seq models in particular, usually employ attention layers [77, 55]. Attention mechanism [13, 45] allows to model dependencies with no regard to their distance in the look-back window. The main idea is to assign soft weights to the hidden states of the encoder to model the relevance of each hidden state to a given query state. Query states are the hidden states of the decoder. This allows to "pay attention" to positions in the input sequence that are most relevant to the target task. A weighted sum is subsequently computed to obtain the corresponding feature [45], as follows:

$$\mathbf{c}_j = \sum_{i=1}^{d} \alpha_{ij} h_i \tag{3.34}$$

where $\mathbf{c}_j$ is the context vector at decoding position j, d is the length of the input sequence, $\alpha_{ij} \in [0, 1]$ is the attention weight that encodes the relevance of encoder hidden state h_j to decoder hidden state h_j. A common weighting function is softmax. More recently, attention mechanism is used in the context of transformers [248] which are increasingly used for natural language processing tasks.

3.4.5 Convolutional Neural Networks

Convolutional neural networks (CNNs) [142] are a class of deep learning models that process data that have grid pattern, like images. This is inspired by the organization of cells in the primate primary visual cortex [112]. CNNs are typically composed of three basic building blocks [92]: a convolution layer, a pooling layer and a fully-connected layer. A convolution is the process of applying *filters* that slide across the time series. The number of elements by which the filter slides at a time is called *stride* (e.g. 1). A filter, also known as *kernel*, performs an element-wise multiplication with the corresponding receptive field at each location of the time series, followed by summing to obtain the output value in the corresponding position, resulting in a feature map. As such, the more convolution layers are stacked, the faster spatial size of feature maps decreases. In some cases, this needs to be avoided using *padding*. Padding is simply adding elements (e.g. zeros) at borders of the original input.

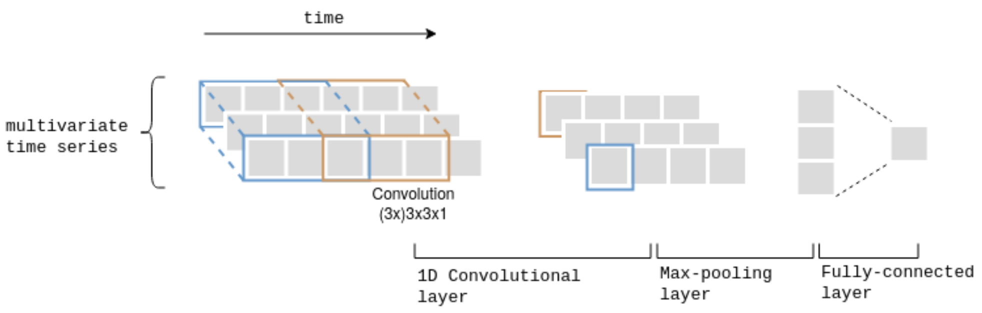
Considering a one-dimensional time series data y on size N and no zero padding, and a one dimensional kernel w of size k, the i-th element of the output feature map is written as:

$$(w * x)(i) = \sum_{j=0}^{k-1} x(i-j)w(j) \tag{3.35}$$

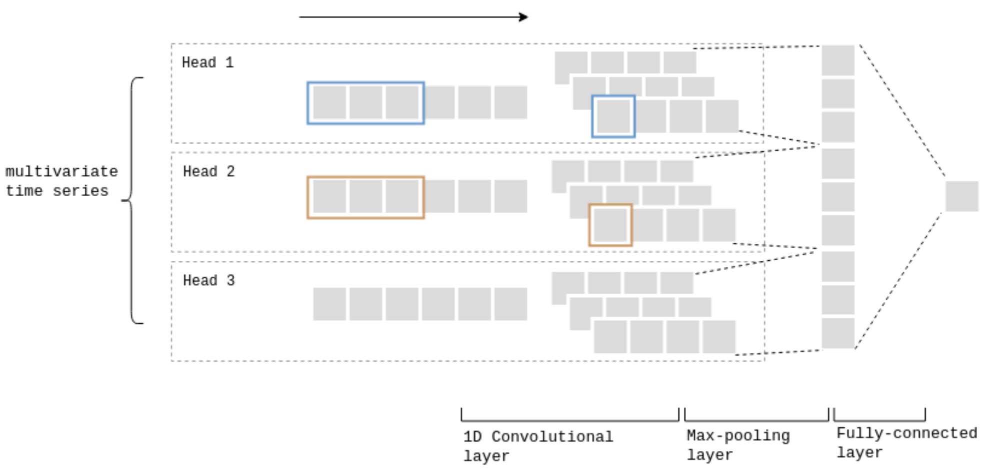
A pooling, such as average or max pooling, takes feature maps as input and reduces their length by aggregating over a sliding window. Max-pooling simply selects the maximum value of each patch of the input data. Instead, average pooling computes the average value. The final fully-connected layers maps the extracted representation of the input time series into final predictions. Note that convolution and pooling layers aim to extract multi-level hierarchical feature representation, while fully-connected layer learn the target objective function (regression or classification).

In convolutional neural networks, each neuron is connected to only a subset of neurons in the previous layer, unlike in the case of FNNs where all neurons are fully connected. This is referred to as *local connectivity* [92]. In addition, all neurons in a particular feature map share the same weights [78]. The same filter or kernel is therefore used at each location. Local connectivity and *parameter sharing* allow to reduce the number of parameters and improve the computational efficiency of learning, especially when dealing with high-dimensional data. Convolutional neural networks have gained a lot of attention with their applications in the fields of computer vision and pattern recognition [132, 132]. Given their success, CNN were adopted to time series analysis tasks [153].

1D Convolutions when processing time series data, filters are one-dimensional (time) instead of two-dimensional (image width and height) [78]. For processing multivariate time series, multi-channel CNNs [270] and multi-head CNNs [42] are used. Both multi-channel and multi-head CNNs separate multivariate time series into multiple univariate time series. In multi-channel CNN [270], each channel corresponds to an univariate time series. Hence, a single feature map is obtained as a final result for all time series. By contrast, multi-head CNN [42] uses a separate sub-CNN model, coined convolution head, for each univariate time series. Hence, an independent feature map is obtained for each time series. Resulted convolutional features are subsequently concatenated together and sent to the fully connected layer for regression [42]. Figure 3.16 illustrates the difference in architecture between multi-channel and multi-head CNNs. Note that in several works, multi-head CNN may be referred to as multi-channel CNN, while a multi-channel CNN is considered as conventional CNN as in [285]. However, we choose the former terminology, to not confuse between channels as existing component of conventional CNN and channels as separate convolution heads.



(a) Multi-channel CNN



(b) Multi-head CNN

Figure 3.16: Illustrations of different architectures of CNN for multivariate time series forecasting: Multi-channel CNN (top) and Multi-head CNN (bottom).

Causal Convolutions refer to the type of convolution that ensures that an output at time *t* is derived only from inputs from time *t* or earlier. This is particularly required for time series forecasting, where information should not be "leaked" from future to past. In one-dimensional convolutions, causal structure is implemented by simply shifting the output to the right direction by a number of time steps [187]. An example of causal convolution in three-layer network is illustrated in Figure 3.17.

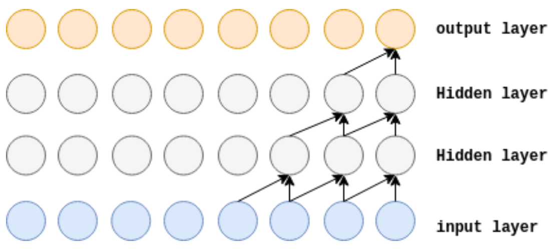


Figure 3.17: An illustrative example of causal convolutional neural network composed of three layers.

Dilated Convolutions (also called "*à trous*") standard convolutions are computationally expensive when dealing with long-term dependencies, as we are increasing the size of the receptive field. Dilated convolutions [274] use "inflated" kernels that allow to access more distant past when forecasting, without requiring additional parameters. This is done by inserting spacing (gaps) between the kernel elements, unlike standard convolution where kernel elements are adjacent. An example of a structure of causal and dilated convolutions in three-layer network is illustrated in Figure 3.18.

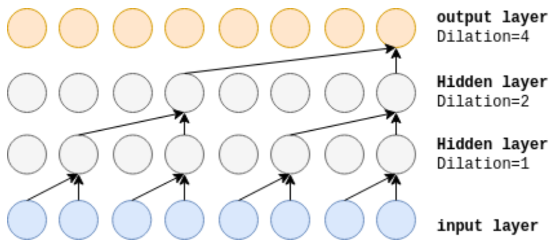


Figure 3.18: An illustrative example of causal and dilated convolutional neural network composed of three layers.

A dilated convolution operation $*_d$ with dilation factor d is written as:

$$(w *_d x)(i) = \sum_{j=0}^{k-1} x(i - dj)w(j) \tag{3.36}$$

In [274], authors show that stacking dilated convolutions with increasingly large d results in expanding receptive fields exponentially. This is particularly useful when modeling long-term dependencies.

An example of model that uses causal and dilated convolutions is deepMind's *WaveNet* [187]. WaveNet was proposed for generating raw audio waveforms.

3.4.6 Comparative Analysis

To summarize, we have presented three techniques that are usually implemented in the context of multi-step multivariate time series forecasting, which are support vector regression, recurrent neural networks and convolutional neural networks. SVR is a "shallow" machine learning model that is time and memory efficient. Unlike neural networks, traditional machine learning algorithms like SVR are interpretable and explainable. However, SVR is generally outperformed by RNNs and CNNs in the context of time series forecasting [73]. SVR also fails to capture long-term dependencies across time series data.

RNNs were proposed for modeling sequential data and capture short and long-term dependencies in sequential data [155].. They shown great results in a variety of application, such as speech recognition, machine translation or image captioning [121] In addition, sequence-to-sequence architecture has been successfully proposed for multi-step ahead time series forecasting [238].

CNNs were initially proposed and intensively investigated in the context of image processing. Motivated by its success, researchers have adopted CNNs for time series forecasting [78]. Intuitively, spatial correlations present in images is analogous to temporal correlations present in time series data. Thanks to the nice local connectivity and parameter sharing properties in CNNs [78, 92], number of parameters of network is considerably reduced, hence a lower memory requirement than RNNs. Given the absence of recurrent connections, CNNs computations can happen fully in parallel. RNNs, on the other hand, need (for the most part) to be processed sequentially [191, 78]. As a result, CNNs computations are generally faster than RNNs'. Furthermore, CNNs are also often better at feature extraction from data, mainly due to position-invariant local patterns [92]. This is particularly useful for transfer learning task which generally require high-level abstract representation learning.

Multi-step ahead forecasting with CNN, however, require some workarounds. One approach consists in combining CNN and RNN [269]. Some works uses CNN for feature extraction from time series data, along with RNN for prediction [269]. Another approach consists simply output a vector instead of one value, while using multi-head or multi-channel CNN [42].

We summarize pros and cons of each of these techniques in Table 3.1.

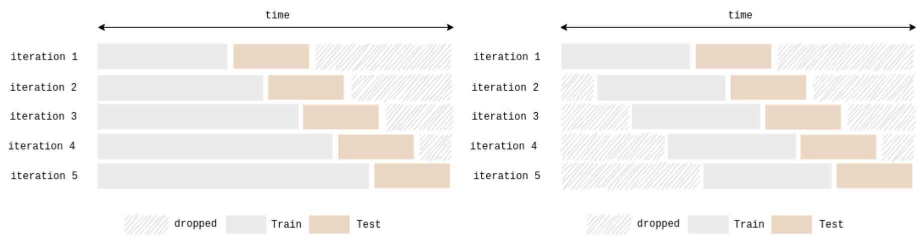
Table 3.1: A summary of the pros and cons of three time series forecasting techniques

Strategy	Pros	Cons
SVR	• Interpretable AI (white box), • Simple, • Computationally cheap and efficient	• Generally outperformed by RNNs and CNNs, • Depend on feature selection
RNN	• Native support for sequential data, • Good performance in short-term and long-term forecasting (ability to capture long-term dependencies)	• Requires large amount of historical data, • Prone to vanishing and exploding gradients issue, • Computationally expensive
CNN	• Fully parallelizable computations, • Good feature extraction ability, • Computational efficiency (no recurrent connections, parameter sharing), • Lower memory requirement (local connectivity and parameter sharing)	• Requires large amount of historical data

3.4.7 Walk-Forward Validation

In classic machine learning tasks, a train-test split consists simply in splitting the data by observation. For example, 80% of observations for training and remaining 20% is held-out for test. The common *k*-fold cross-validation scheme was previously illustrated in Figure 3.5. This mainly relies on the assumption that observations are independent. However, this assumption is violated in case of sequential data. When performing classic cross-validation on tile series data, it makes no sense to mix the order of the observations give that the model will get to peek into the future. This is sometimes referred to as *data leakage*. Data leakage usually results in overly optimistic estimated performance of trained model.

Walk-forward validation is cross-validation equivalent for time-series data. Two major approaches exist: *expanding window* (also called *forward chaining*) and *sliding window*. Expanding window consists in training the model on all available historical data, whereas sliding window consists in training the model on only most recent observations. Figure 3.19 gives a visual illustration of both walk-forward approaches.



(a) Expanding window **(b)** Sliding window

Figure 3.19: Walk-forward validation strategies for time series data.

3.4.8 Evaluation Metrics for Time Series Forecasting

Performance evaluation involves assessing the accuracy of predictive models and consists on a primordial step of the machine learning pipeline. Different set of metrics are chosen depending on the learning task, such as regression, classification, clustering, etc., and the specific requirements of the problem to solve. In the context of our work, we are dealing with time series forecasting. Common regression primary metrics [35] used to assess prediction models are; Mean Absolute Error (MAE), Mean Bias Error (MBE), Root Mean Squared Error (RMSE), etc.

MAE measures the average magnitude of the errors in a set of predictions, using the formula:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i| \, ,$$

where y_i and $\hat{y}_i$ respectively denote the true value of the predicted value of the i-th data sample, and N denotes the size of the dataset. If the absolute value is not taken, the MAE becomes the MBE. MBE however suffers from a cancellation effect, where positive bias and negative bias cancel out. RMSE is defined as the square root of the average squared distance between prediction and ground truth, using the formula:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

These primary metrics are often extended with additional normalization. The Normalized RMSE (NRMSE) is for example defined by the normalization of the RMSE either by dividing it by the standard variation of the ground truth data, by the difference of the maximum and the minimum values of the ground truth data, or by the mean. The latter normalization by the mean is also known as Coefficient of Variation of Root Mean Square Error (CVRMSE). The Normalized Mean Error (NME) involves the normalization of the MAE by the sum of the ground truth values, and is defined as follows:

$$NME = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{\sum_{i=1}^{N} \hat{y}_i}$$

An example of relative error metric for time series forecasting is mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD). MAPE computes the average absolute percentage errors of forecasts. MAPE is written as follows:

$$MAPE = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

MAPE treats equal errors above the ground truth value and below the ground truth value differently. It comes down to the fact that forecasts that are greater than the ground truth are penalized heavier than forecasts that are lower than the ground truth [11]. To mitigate this issue, symmetric MAPE (sMAPE) was proposed:

$$SMAPE = \frac{100}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{|y_i| + |\hat{y}_i|}$$

Furthermore, MAPE produces infinite or undefined values when the actual values are zero or close to zero, which is a common occurrence in energy consumption field. If the actual values are very small (usually less than one), MAPE yields extremely large percentage errors (outliers), while zero actual values result in infinite MAPEs.

Other composite metrics are also used in practice by combining one or more primary metrics. Coefficient of determination, also denoted as R^2, measures the proportion of variance explained by the prediction model to the total variance in the observed data. R^2 normally ranges between 0 and 1, with 1 indicating the perfect fit. Values may, however, fall outside the range of 0 to 1 if the predictive model is worse than using an horizontal hyperplane that passes through the mean value. R^2 is defined using the formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2},$$

where $\bar{y}$ is the mean of the ground truth data. Authors in [148] considered that R^2 is biased, insufficient and misleading. Thus, it should not be used to assess the accuracy of predictive models.

3.4.9 Similarity Measures for Time Series Data

Time series forecasting is crucial for decision making in several domains, namely building energy modeling. In several cases, it is important to study the similarity between time series data in order to improve forecasting performance [17].

Dynamic time warping (DTW) [180] is one of the most popular distance measures used for time series data. DTW was originally proposed for speech recognition applications to compare disparate speech patterns [180]. Classical distance measures such as Euclidean and Manhattan distances are not suitable when dealing with time series data due to time distortions and time shifts. By contrast, DTW is able to measure similarity between two time series that may differ in duration and speed by aligning them. As DTW suffers from quadratic complexity, several algorithms are proposed to speed it up, such as FastDTW [220] and PrunedDTW [227]. Wang et al. [251] compared several distance metrics and demonstrated that no other metric significantly outperforms DTW.

Izakian et al. [114] proposed a time series clustering framework based on DTW. Fawaz et al. [43] used DTW to quantify the similarity between source and target dataset in their transfer learning framework.

3.5 Summary and conclusions

Machine learning (ML) allows systems to improve automatically with experience using data. Once thoroughly defined, a classical ML process generally starts with collecting relevant data in a sufficient amount for the target task. This is an important step for a successful ML project. ML has demonstrated great success in learning complex tasks and delivering predictions that guide decision making. Decision making became more insightful and accurate with the emergence of deep learning.

In this work, we are particularly interested in the task of building energy modeling (BEM). Historical data on building energy consumption are time series. Consequently, BEM using machine learning relies on time series forecasting techniques. In addition to building energy field, time series forecasting is becoming very important in a broader range of real-world applications, such as healthcare and finance [153]. Most used machine learning techniques for this task are support vector regression (SVR) and neural networks (NN), such as convolutional neural networks (CNN) and recurrent neural networks (RNN).

Compared to NN, SVR is simple and computationally cheap [73]. However, it is generally outperformed by NN. RNN is widely used in the context of time series forecasting due to its ability to capture short and long term dependencies [121, 155]. CNN is computationally efficient and is characterized by its powerful feature extraction capabilities [92]. RNN and CNN require, on the other hand, a large amount of training data compared to SVR.

To improve time series forecasting results, it is sometimes required to cluster or classify available data. Most popular similarity measure for time series data is dynamic time warping (DTW). Existing research works show that DTW is not significantly outperformed by any other well-known distance metrics in the context of time series processing [251].

Training accurate models mainly rely on collected dataset. The main factors to study the suitability of the dataset are quantity and respresentativeness. A relatively large amount of historical data must be collected beforehand. More particularly, in the context of building energy modeling, reported works usually rely on multiple months to years of historical data [127, 22]. Representativeness refers to the relevance of available data to the target task. Namely, for BEM, labeled data collected from the building to model are generally required to reliably model its energy consumption. However, in many cases, buildings do not yet provide historical data, such as newly built and newly renovated buildings.

To alleviate data unavailability challenge, transfer learning is usually required. This allows to reuse available existing data or existing models for the training of models for an unseen target. A detailed description of this area of research is in Chapter 4.

Chapter 4

Cross-Domain Knowledge Transfer

4.1 Introduction

Traditional machine learning applications usually assume that the training and the test data are in the same feature space and are drawn from the same distribution. However, in many real-life applications, this assumption does not hold. For example, in the field of building energy modeling, energy consumption is complex and is heavily affected by a wide range of variables. As such, energy profiles may vary greatly with small changes in such variables (e.g. renovation, different operation schedules, change of tenants, varying weather conditions, etc.). Similarly, in computer vision, photos and sketches do not follow the same distributions. To mitigate this issue, the *transfer learning* (TL) area of research was introduced. TL is also referred to in literature as *knowledge transfer*.

This chapter is structured as follows. Section 4.2 introduces the main field of transfer learning from which derive domain adaptation and domain generalization sub-fields. Section 4.3, Section 4.4 and Section 4.5 provides an overview of various approaches proposed in the context of respectively domain adaptation, multi-source domain adaptation and domain generalization. Section 4.6 summarizes different similarity metrics usually used for transfer learning.

4.2 Overview on Transfer learning

The notations and definitions provided in this section follow the survey paper by Pan et al. [190]. The definition of TL relies on two main concepts, the *domain* and the *task*. A *domain* $\mathcal{D}$ is composed of a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where $X = \{x_1, x_2, ..., x_n\} \in \mathcal{X}$ is a set of examples of size n. A domain is denoted by $\mathcal{D} = (\mathcal{X}, P(X))$. If two domains are different, then their feature spaces are different and/or their marginal probability distributions are different. For a particular domain $\mathcal{D}$, a *task* $\mathcal{T}$ is composed of a label space $\mathcal{Y}$ and an objective function $f(.)$

that is learned from the training data pairs (x_i, y_i), where $x_i \in \mathcal{X}$ is the feature vector of example i and $y_i \in \mathcal{Y}$ is its corresponding label. The objective function $f(.)$ is used to predict a label y of each new example x. From a probabilistic viewpoint, $f(.)$ consists in the conditional probability $P(y/x)$. A task is denoted by $\mathcal{T} = (\mathcal{Y}, f(.))$. We consider a source domain $\mathcal{D}_S$ and a target domain $\mathcal{D}_T$. The source domain data are denoted as $\mathcal{D}_S\{(x_{S_1}, y_{S_1}), ..., (x_{S_n}, y_{S_n})\}$, where the data instance $x_{S_i} \in \mathcal{X}_S$ and its corresponding label $y_{S_i} \in \mathcal{Y}_S$. Similarly, the target domain data are denoted as $\mathcal{D}_T\{(x_{T_1}, y_{T_1}), ..., (x_{T_n}, y_{T_n})\}$, where $x_{T_i} \in \mathcal{X}_T$ and $y_{T_i} \in \mathcal{Y}_T$.

More precisely, for the task of building energy modeling, energy consumption depends greatly on several building-specific contextual characteristics, namely typology (residential, industrial, or commercial), shape, size and age [264]. Hence, buildings have varying probability distributions, over different characteristics. Here, each building setting may be considered as a disparate domain with a different distribution.

Transfer learning is defined as [190]: "Given a source domain $\mathcal{D}_S$ and a learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and a learning task $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target objective function $f_T(.)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$."

Given that a domain consists of the pair $\mathcal{D} = (\mathcal{X}, P(X))$, the expression $\mathcal{D}_S \neq \mathcal{D}_T$ implies that (1) the feature spaces between the source and the target domain are different, i.e. $\mathcal{X}_S \neq \mathcal{X}_T$, or (2) the feature spaces are the same but the marginal probability distribution between the source and the target domains are different, i.e. $P_S(X) \neq P_T(X)$. Similarly, a task consists of the pair $\mathcal{T} = (\mathcal{Y}, f(.))$, the expression $\mathcal{T}_S \neq \mathcal{T}_T$ implies that (1) the label spaces between the source and the target domains are different, i.e. $\mathcal{Y}_S \neq \mathcal{Y}_T$, or (2) the label spaces are the same but the objective functions between the source and the target domains are different, i.e. $f_S(.) \neq f_T(.)$. A traditional machine learning problem is thus characterized by the same source and target domains , i.e. $\mathcal{D}_S = \mathcal{D}_T$ and the same source and target tasks, i.e. $\mathcal{T}_S = \mathcal{T}_T$.

Authors in [190] first classify transfer learning techniques into three settings as depicted (see Figure 4.1): (1) inductive transfer learning, (2) transductive transfer learning and (3) unsupervised transfer learning. *Inductive transfer learning* refers to the case where the target task is different from the source task. Labeled data are however required in the target domain to be able to train the target model. Inductive TL models learn from labeled/unlabeled data from the source domain and labeled data from the target domain. Therefore, inductive transfer learning setting is categorized into various sub-settings, depending on whether the source domain data are labeled or unlabeled, and depending on how target and source tasks are learned. Namely, multi-task learning refers to the case where a lot of labeled are available in the target domain and source and target tasks are learned simultaneously [190]. By contrast, sequential transfer learning refers to the case where source and target tasks are learned sequentially. Self-taught learning refers, on the other hand, to the case where no labeled data are available in the source domain.

Unsupervised transfer learning setting is similar to inductive transfer learning in the sense that the target task is different from the source task. However, unsupervised transfer learning is related to unsupervised learning tasks, such as clustering, dimensionality reduction and density estimation, where no labeled data are available in both the source and the target domains.

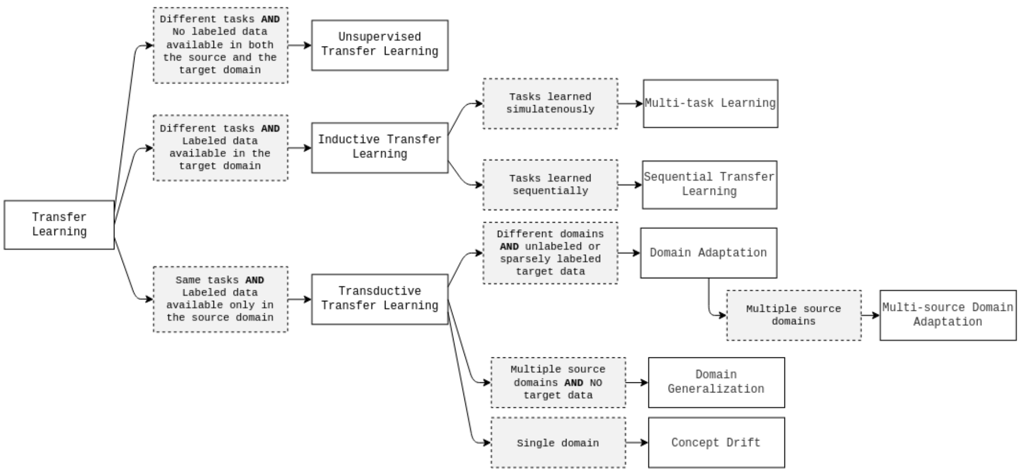
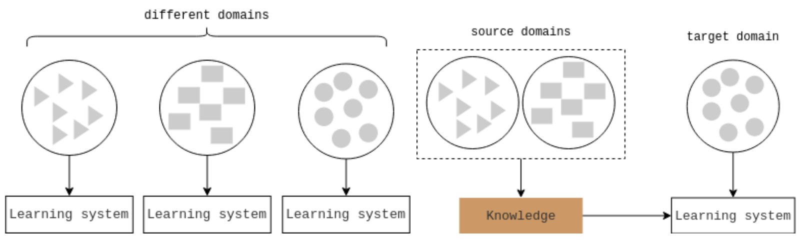

Transfer Learning
Different tasks AND No labeled data available in both the source and the target domain
Unsupervised Transfer Learning
Different tasks AND Labeled data available in the target domain
Inductive Transfer Learning
Tasks learned simulatenously
Multi-task Learning
Tasks learned sequentially
Sequential Transfer Learning
Same tasks AND Labeled data available only in the source domain
Transductive Transfer Learning
Different domains AND unlabeled or sparsely labeled target data
Domain Adaptation
Multiple source domains
Multi-source Domain Adaptation
Multiple source domains AND NO target data
Domain Generalization
Single domain
Concept Drift

Figure 4.1: Overview of different settings of transfer learning. Dotted rectangles indicate assumptions. Rectangles indicate transfer learning sub-classes. Adapted from [190].

Transductive transfer learning setting consists in having the same target and source tasks, but different target and source domains. In transductive TL setting, we assume that we have a relatively large amount of labeled data in the source domain.Typically, more data are available in the source domain than in the target domain. In this work, we focus on two main sub-settings of transductive transfer learning, which are domain adaptation and domain generalization. *Domain adaptation* (DA) fits the scenario where labeled data are available in the source domain and unlabeled or few labeled data are available in the target domain. *Multi-source domain adaptation* (MDA) is an extension of DA in which we have access to multiple source domains.

Domain generalization (DG) fits the scenario where multiple source domains are available, but no data are required in the target domain. In other words, DG assumes no prior knowledge on the target domain during training, and therefore aims to build models that can directly generalize out-of-the-box to any new domain. Arguably, domain generalization is a harder problem than domain adaptation.

Broadly, four different transfer learning approaches are to be distinguished [190]: (1) instance-based transfer, (2) feature-representation-based transfer, (3) model-based transfer, and (4) relational-knowledge-based transfer. Instance-based transfer learning consists in re-weighting labeled instances of the source domain and re-use them in the target domain. Feature-representation-based transfer learning seeks to find a common



(a) Traditional machine learning **(b)** Domain adaptation

Figure 4.2: Different learning processes between traditional machine learning and domain adaptation [190]

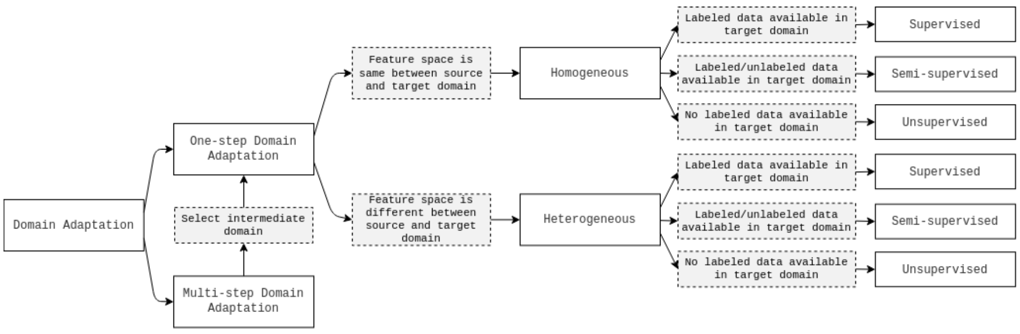
feature representation of the source and target domain, that reduces domain discrepancy. Model-based approaches aim to discover shared parameters or priors between models of the source and the target domains. Relational-knowledge-based methods build a relational knowledge mapping between the source and the target domains. Generally, for DA and DG, instance-based, feature-representation-based and model-based approaches are proposed.

4.3 Domain Adaptation

4.3.1 Taxonomy of Domain Adaptation settings

Domain adaptation can be categorized into two main classes; homogeneous and heterogeneous DA. In the case where the feature representations for the target and the source domains are the same $X_T = X_S$, we refer to the problem as *homogeneous DA*. Consequently, it is the data distributions that differ between the target and the source domains $P_T(X) \neq P_s(X)$. In the case where the feature representations for the target and the source domains $X_T \neq X_S$, we refer to the problem as *heterogeneous DA*. The above mentioned classes may be further split into supervised, semi-supervised, and unsupervised DA. In *supervised DA*, we have a small amount of labeled data in the target domain. In *semi-supervised DA*, we have both labeled and unlabeled data in the target domain. In *unsupervised DA*, we have no labeled data in the target domain. This classification is depicted in Figure 4.3. It is also essential to note that in some works [62], the unsupervised DA that we defined here is rather considered as a *semi-supervised DA* given that we have labeled data in the source domain and unlabeled data in the target domain. The *unsupervised DA* naming in this case refers to the domain adaptation unsupervised learning problems such as clustering, dimensionality reduction, etc. For what follows, we will consider the former classification.

The approach proposed by Wang et al. in [250] distinguished one-step DA and multi-step DA. In one-step DA, the knowledge transfer is directly accomplished in one-step between the source and the target domains. By contrast, in multi-step DA, we seek to identify intermediate domains that play the role of bridges that connect seemingly



Domain Adaptation
One-step Domain Adaptation
Select intermediate domain
Multi-step Domain Adaptation
Feature space is same between source and target domain
Feature space is different between source and target domain
Homogeneous
Heterogeneous
Labeled data available in target domain
Labeled/unlabeled data available in target domain
No labeled data available in target domain
Supervised
Semi-supervised
Unsupervised

Figure 4.3: Overview of different settings of domain adaptation [250].

unrelated source and target domains. Multi-step DA [242] considers the distance between domains.

We distinguish three main approaches to domain adaptation, instance-based approaches, feature representation-based approaches and model-based approaches. A comprehensive review of domain adaptation can be found in [278]. Instance-based adaptation assumes that certain data points of the source domain are more similar to the target domain than others, and thus seek to reweigh instances correspondingly; similar instances will have stronger weights. Feature-representation-based methods assume that there exist a feature space in which source and target features are indistinguishable. These methods aim to find this feature space. Model-based adaptation aim to adapt models trained on the source domain in order to perform well on the target domain.

4.3.2 Early Approaches of Domain Adaptation

Instance re-weighting methods These methods aim to re-weight or select data instances that reduce the discrepancy between source and target domains. Instance reweighting schemes are well documented in literature. As seen in Chapter 3, given a data set and a hypothesis space $\mathcal{H}$, a machine learning algorithm might aim to learn the optimal hypothesis $f^* \in \mathcal{H}$ that minimizes the expected risk, such that:

$$f^* = \arg\min_{f \in \mathcal{H}} \mathbb{E}_{(x,y) \in \mathcal{X} \times \mathcal{Y}}[\mathcal{L}(f(x), y)]$$

where $\mathcal{L}$ is the loss function, $\mathcal{X}$ is the input space, and $\mathcal{Y}$ is the output space.

Ideally, in the domain adaptation setting, we want to learn the optimal model for the target domain that minimizes the empirical risk over the target distribution:

$$
\begin{aligned}
f_T^* &= \arg\min_{f \in \mathcal{H}} \mathbb{E}_{(x,y) \in \mathcal{X} \times \mathcal{Y}} [\mathcal{L}(f(x), y)] \\
&= \arg\min_{f \in \mathcal{H}} \sum_{(x,y) \in \mathcal{X} \times \mathcal{Y}} [P_T(x,y) \mathcal{L}(f(x), y)]
\end{aligned}
$$

However, since no labeled data are generally available in the target domain, we learn the optimal model on the source data instead. If the target and the source data follows the same distribution, we can simply learn an optimal model on the source data. The learned model may be directly used in the target domain without further optimization:

$$
f^* = \arg\min_{f \in \mathcal{H}} \mathbb{E}_{(x,y) \in \mathcal{X} \times \mathcal{Y}} \mathcal{L}(f(x), y)
$$

Otherwise, when the probability distributions between the target and the source domains differ, we rather need to consider the following optimization problem:

$$
\begin{aligned}
f_T^* &= \arg\min_{f \in \mathcal{H}} \sum_{(x,y) \in \mathcal{X} \times \mathcal{Y}} \frac{P_S(x,y)}{P_S(x,y)} P_T(x,y) \mathcal{L}(f(x), y) \\
&= \arg\min_{f \in \mathcal{H}} \sum_{(x,y) \in \mathcal{X} \times \mathcal{Y}} \frac{P_T(x,y)}{P_S(x,y)} P_S(x,y) \mathcal{L}(f(x), y) \\
&\approx \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{N_S} \frac{P_T(x_{S_i}, y_{S_i})}{P_S(x_{S_i}, y_{S_i})} \mathcal{L}(f(x_{S_i}), y_{S_i})
\end{aligned}
$$

Hence, we can learn an accurate model for the target domain by weighting the importance of each instance (x_{S_i}, y_{S_i}) of the source domain by assigning a weight $\frac{P_T(x_{S_i}, y_{S_i})}{P_S(x_{S_i}, y_{S_i})}$. Note that for approach, we assume a *homogeneous* DA setting. Given that we generally do not have labeled data in the target domain, the density ratio $\frac{P_T(x_{S_i}, y_{S_i})}{P_S(x_{S_i}, y_{S_i})}$ is difficult to compute. As a consequence of Bayes' rule, two research ways have been proposed in literature to tackle this challenge. In the first approach, we assume that the source and the target domains have the same conditional distribution of $\mathcal{X}$ $P_S(x/y) = P_T(x/y)$ and different class distributions $P_S(y) \neq P_T(y)$. The different class distributions between the two domains is usually referred to as *target shift* [277], *prior probability shift* [201] or *class imbalance* [117]. In the second approach, we assume that the source and the target domains have the same conditional distribution $\mathcal{Y}$ $P_S(y/x) = P_T(y/x)$ and different marginal distributions of $\mathcal{X}$ $P_S(x) \neq P_T(x)$. The different between the two domains is referred to as *covariate shift* [201] or *sample selection bias*.

Under covariate shift, the ratio between the source and the target domains joint distributions is rewritten as:

$$\frac{P_T(x,y)}{P_S(x,y)} = \frac{P_T(x)}{P_S(x)}$$

Therefore, we compute importance weights by estimating the ratio of the data marginal distributions $\frac{P_T(x)}{P_S(x)}$. As such, instances with high ratio are more important to the target domain than instances with low ratio. The problem of estimating importance weights is also referred to as density ratio estimation (DRE). Several DRE methods have been proposed in literature, such as kernel density estimation, kernel mean matching, and the use of probabilistic classifiers.

Indirect DRE methods estimate the marginal distribution of each domain independently and subsequently take their ratio. By contrast, direct DRE methods directly estimate weights by minimizing the discrepancy between the source and the target domains (without density estimation). A popular domain discrepancy measure is maximum mean discrepancy (MMD) [94, 189, 111]. Minimizing MMD in a high-dimensional space, namely a reproducing kernel Hilbert space (RKHS), is called kernel mean matching (KMM) [111]. KMM methods can be successfully used to re-weight individual instances. However, their time complexity is cubic in the size of training data, which is computationally impractical on large data [131].

Another direct DRE is through Kullback-Leibler Importance Estimation Procedure (KLIEP) [234]. KLIEP uses KL-divergence as domain discrepancy measure. The major benefit of KLIEP is that it can estimate the values of the importance of new target instances without further weight estimation, as opposed to KMM [131]. Another simple method transforms the DRE problem into a binary classification problem; by trying to predict whether each instance is coming from the source or the target domain data [276, 26]. As such, importance weights are computed by inverting posterior probabilities.

Under prior probability shift, the ratio between the source and the target domains joint distributions is rewritten as:

$$\frac{P_T(x,y)}{P_S(x,y)} = \frac{P_T(y)}{P_S(y)}$$

Therefore, we compute importance weights by the estimation of $\frac{P_T(y)}{P_S(y)}$. Note that in this case, it consists in weighing the importance of domain labels rather than data instances [154].

Feature representation-based methods These approaches argue that there exists a common feature space that minimizes discrepancy between source and target domains. Existing works in this context may be classified into three approaches [278], namely feature transformation [62], subspace learning [279, 81], feature reconstruction [118] and feature coding [266]. A detailed review of domain adaptation approaches can be found in [278].

Model-based methods Model-based methods aim to adapt models trained on the source domain in order to perform well on the target domain. Generally, these methods require at least a small set of labeled target data, therefore they can be applied only to the supervised or the semi-supervised DA scenarios [60]. Yang et al. [271] proposed an adaptive SVM (A-SVM) for adapting source classifiers to a target domain. Authors also propose a performance evaluation metric that enables to select the best source classifiers for adaptation.

4.3.3 Deep domain Adaptation

Deep learning [92] has demonstrated its powerful capability to learn highly salient representations from raw data. These representation usually perform better than traditional hand-engineered features in many fields. However, fully-supervised deep networks trained on limited data would often dramatically over-fit [70]. Domain adaptation thus seeks to exploit deep learning advances to learn feature representations that are robust to domain shift. Deep domain adaptation approaches can be classified under three axes [60]: (1) using deep models to extract deep representations that can be then used by shallow domain adaptations methods, (2) train a deep network on the source domain and later further fine-tune it to the target domain, and (3) use a deep network architecture that is specifically designed to domain adaptation. Work presented by Wang et al. in [250] further categorized the latter axis into three approaches: (1) *Discrepancy-based* approaches that aim to minimize the domain shift by fine-tuning deep networks using labeled or unlabeled target data, (2) *Adversarial-based* approaches that promote domain confusion using domain discriminators through adversarial objectives, and (3) *Reconstruction-based* approaches that enforce feature domain invariance using data reconstruction as auxiliary task.

Adversarial-based approaches is further categorized into generative and non generative methods. *Generative* methods typically aim to use source domain data, noise vectors, or both to generate samples that are similar to target domain. These simulated samples along with annotations from source domain will constitute training data. *Non-generative* methods learn discriminative feature representations using labels in the source domain, and later map target data to the same feature space through a domain-confusion loss. Similarly, reconstruction-based approaches are categorized into encoder-decoder and adversarial reconstruction methods. *Encoder-decoder* reconstruction methods use an encoder for representation learning, and a decoder for data reconstruction. *Adversarial* reconstruction methods are based on reconstruction error between original and reconstruction example in each domain [250]. Table 4.1 summarizes one-step deep DA approaches as presented in [250].

Table 4.1: One-step deep domain adaptation approaches. Adapted from [250].

Deep DA approach	Description	Sub-settings
Discrepancy-based	minimize the domain shift by fine-tuning deep networks using labeled or unlabeled target data	class-criterion, statistic criterion, architecture criterion, geometric criterion
Adversarial-based	promote domain confusion using domain discriminators through adversarial objectives	generative, non-generative
Reconstruction-based	enforce feature domain invariance using data reconstruction as auxiliary task	encoder-decoder [288], discriminative

Zhuang et al. [288] proposed a supervised deep encoder-decoder for learning domain-invariant feature representations. Taigman et al. [241] proposed a generative adversarial network (GAN)-based for unsupervised DA. The main idea is to generate data in the target domain by transforming source domain data (to seem as if drawn from target domain).

4.4 Multi-source Domain Adaptation

Above mentioned methods were designed for cases where we have a single source domain and a single target domain. By contrast, multi-source domain adaptation [166, 236] deals with source data collected from different sources. A straightforward approach consists in concatenating different available source domains into a single source set. We refer to this approach as *source-combined DA*. However, this approach might yield a poor performance because of the domain shift between the multiple source domains. Because domain shift not only exist between each source and the target but also among multiple source domains, models trained on source-combined data from disparate sources can interfere with one another during the training process and generalize poorly when applied to the target domain [283, 207].

Assuming that the target distribution can be represented as mixture of multi-source distributions, Mansour et al. [165] proposed a distribution weight combining rule that combines source models and uses as weights mixture coefficients. Xu et al. [267] proposed a deep cocktail network, inspired by [165], that uses multiple domain discriminators to achieve latent space alignment. Zhu and al. [287] proposed a two-stage alignment framework, that aligns domain-specific distributions of each source-target domain pair and aligns outputs of domain-specific classifiers for each target sample.

Another approach for MDA is through *source domain selection*. Source domain selection [74, 49] mainly consists in selecting source domains that are the most pertinent to the target domain, or assigning a weight to each source domain depending on its similarity to the target domain. In other words, it consists in selecting source domains that holds relevant information to transfer, and discarding the ones that might negatively impact the target model [60]. Duan et al. [74] proposed a domain selection machine that is trained on loosely labeled web images from disparate source domains. Proposed domain selection relies on a weighted combination of source classifiers as well as a domain-dependent regularizer for selecting most pertinent source domains. Chen et al. [49] proposes a re-weighting vector to match the source domain label distribution to the unknown target one. Bhat et al. [24] propose to select the best source domains based on both $\mathcal{H}$-similarity and complementary properties, and then to iteratively learn a shared representation.

Generally, one source dataset is considered as one domain. For instance, images collected from ImageNet, from Caltech and from Bing are considered to represent three source domains. However, this assumption does not always hold, because domain shift may exist among data from one source. Several works have thus proposed to discover latent domains in multi-source data [105, 265, 164].

Multi-source DA is sometimes referred to as multi-domain learning (MDL) [272]. Yang et al. [272] propose a unified framework that tackles both multi-domain learning (MDL) and multi-task learning (MTL) problems, where we deal with respectively multiple source domains and multiple learning tasks. The main idea is to exploit available *semantic descriptors* (or metadata) (e.g. indexes of domains or tasks) about each feature vector. As such, the need to distinguish between different domains and different tasks is alleviated. Their proposed network is two-sided; one performs representation learning from feature vector, and one from associated semantic descriptor. Both sides outputs are subsequently combined into a single output. This work can be further applied for zero-shot domain adaptation (ZSDA) (or domain generalization) setting which assumes that *no* target data are available during training-time.

4.5 Domain Generalization

Domain generalization (DG) approach tackles the domain shift problem as in DA. DG addresses building models that by design generalize well even in *new previously unseen* target domains. DG is related to the multi-domain learning (MDL), in the sense that it also aims to learn a single model that is effective for multiple known domains. However, in contrast to MDL, DG models can be generalized to new unknown target/testing domains. In literature, DG may be investigated under different names, i.e. zero-shot domain adaptation (ZSDA) and zero-shot learning (ZSL) [272]. Note that several works use the aforementioned names to describe an unsupervised domain adaptation setting, where training *unlabeled* target data are available [29]. ZSL naming is also often used to describe multi-task learning setting, e.g. recognizing new classes.

Domain generalization approaches proposed in literature may be roughly classified into three axis: (1) data representation based approaches, (2) ensembling approaches and (3) meta-leaning approaches. Our taxonomy of domain generalization approaches is summarized in table 4.2.

Table 4.2: Our taxonomy of domain generalization approaches.

Approaches	Description	Reference
Representation-based	aim to learn a generalized data representation for unseen target domains. (1) several works propose to learn a domain-invariant feature representation that minimizes the domain discrepancy between multiple source domains, (2) while others rely on the assumption that a domain is composed of a domain-specific and a underlying domain-agnostic parts. The goal is hence to learn to extract the domain-agnostic part so that knowledge can be transferred to the unseen target domains	[179, 90, 37, 178, 147, 151, 254], [125, 68, 145]
Ensembling	training domain-specific models for each source domain, and then optimally fuse them at test time.	[268, 163]
Meta-learning	present model agnostic training strategies to train more robust models to domain shift	[146, 16, 152, 71]

Given multiple source datasets, Khosla et al.[125] proposed an SVM based approach, in which the learned weight vectors are common to all datasets. Muandet et al. [179] proposed to learn new domain-invariant feature representations by minimizing the dissimilarity across domains via domain-invariant component analysis and a kernel-based optimization algorithm. Ghifary et al. [90] proposes a multi-task autoencoder MTAE that extends autoencoders into a model that jointly learns to perform self-domain data reconstruction and between-domain data reconstruction. Li and al. in [147] proposed adversarial autoencoders with Maximum Mean Discrepancy (MMD) measure as domain-distance regularization to learn a universal feature representation across domains. Authors propose to jointly minimize reconstruction error, classification error and domain difference while imposing a prior distribution in the learned feature space via adversarial training. Li et al. [145] proposed a low-rank parameterized convolutional neural network model for end-to-end DG learning.

Xu et al. [268] trained low-rank exemplar-SVMs, which can be defined as a linear SVM classifier trained on a single positive training instance and all negative training instances, for both domain adaptation and domain generalization. For domain generalization, the authors propose to whether equally fuse all exemplar classifiers, or use the exemplar classifiers in the latent domain which the target data more likely belongs to. Mancini et al. [163] proposed to train domain-specific classifiers, one on each source domain. Classifiers are then optimally combined at test time based on similarity between target image and source images.

Li et al. [146] propose a meta-learning domain generalization approach MLDG.

It consists on a model agnostic training procedure that can improve the domain generality of a base learner. This procedure is based on synthesizing virtual training and virtual testing domains within each mini-batch. The meta-optimization objective consists on minimizing the loss in the training domains, while simultaneously improving the virtual testing loss. Balagi et al. [16] propose a scheme for learning regularization functions that generalize to novel distributions.

Mancini et al. [162] propose to use *target domain metadata* and model domain dependencies using a graph in the context predictive domain adaptation. Authors in [162] propose to build multiple domain-specific models on each source domain, and then regress a model for the target domain based on nearby domains in the graph. Domain metadata for transfer learning was also explored by Yang et al. [273]. Yang et al. [273] proposed a cross-domain multivariate regression approach by mapping from domain metadata to points in a Grassmanian manifold. Given this mapping and target domain metadata, two solutions are proposed to infer a classifier in the unseen target domain.

4.6 Similarity Measures between Domains

For a successful generalization across multiple domains, a domain similarity measure is necessary to compute discrepancy between data distributions. Numerous metrics are proposed in literature, such as maximum mean discrepancy [33, 158, 147, 287], Kullback-Leibler divergence [234, 241], f-divergence [186], Wasserstein metric [226], Kolmogorov-Smirnoff statistic [131], or Rényi divergence [56]. However, these metrics require at least unlabeled data from the target domain to compute similarities.

Several works choose to rather learn a good similarity metric that fits specific requirements. For instance, Pinheiro et al. [195] proposed an unsupervised DA approach based on similarity learning. The authors aim to find a representation space in which instances of a given category (independently of their domains) are mapped around this category's prototype. A prototype of a category is computed by averaging representations of all data instances that belong to this category. At test time, the most similar prototype to a target sample is selected.

A detailed review on metric learning from data can be found in [20]. In general, similarity metric learning [134] aims to learn a task-specific distance metric using similarity information delivered by training data. This is needed when general-purpose distance metrics (e.g. Euclidean distance or cosine similarity) fail to capture data characteristics and relationships [20]. It is also required when usual distance metrics are not well-suited for particular tasks and requirements. The simplest and standard approach for learning similarity metrics is to find a linear transformation on data (often called Mahalanobis metric learning [82]), so that similar data points are brought closer together while dissimilar data points are pushed farther apart.

More recently, non-linear metric learning is used to capture non-linear characteristics within data. Most existing works in this framework are inspired from Siamese networks [122] due to their powerful representation capabilities [212]. A Siamese network [40, 52] contains two identical sub-networks. During training, it takes two inputs, one in each sub-network, and learns to predict similarity between these inputs. Maotiian et al. [178] used a Siamese network in the context of visual supervised domain adaptation and generalization. Their work aims to find a shared embedding subspace for source and target distributions that promotes both domain confusion and semantic alignment. Semantic alignment ensures that data points from different domains but similar labels are mapped close together in the learned embedding subspace.

4.7 Summary and conclusions

To summarize, we categorized the field of transfer learning, and presented shallow and deep approaches of domain adaptation. DA setting is characterized by having the same learning task but different domains across the source and the target settings. In contrast to DA, domain generalization deals with scenarios where *no data* are available in the target domain. The goal is therefore to study how a model accurately performs out-of-the box in new domains [146].

Generally, cross-domain knowledge transfer applications assume that the source and the target domains are related, and seek to adapt data instances, feature representations or models from the source domains to the target domain. However, this is not always the case. In real-world large-scale scenarios, many source domains are available and are potentially very dissimilar. In consequence, applying techniques that were proposed for relatively related domain may not hold good generalization performance.

Some works proposed to select most similar source domains to the target domain. For studying similarity across different domains, distance metrics generally rely on the availability of target data. However, target data are not always available. This is particularly the case in newly built or newly renovated buildings, in which no operational data are preliminarily generated.

Most of TL research was interested in fields like computer vision and natural language processing. TL has therefore not been thoroughly investigated for time series forecasting tasks, which are useful across various applications such as energy, finances, network traffic or healthcare.

Part 2

Scientific Contributions

Chapter 5

Query-adaptive Training Data Recommendation

5.1 Introduction

Machine learning is increasingly and successfully used in many sectors and industries. However, successful machine learning application depends heavily on the availability of sufficiently large amount of relevant data, notably when deep learning is used. Training accurate models when no data are available in the target domain (for which the model is intended), is therefore challenging.

With the growth of open data initiatives in many fields, existing data about other related domains may be accessible. The main idea of our work is, thus, to leverage data collected from multiple different and related source domains when *no data* are available on a target domain. An important challenge arises when handling multi-source data, as large domain shift may exist between different source domains, as well as between each source domain and target domain [235]. Consequently, models trained on source-combined data collected from disparate sources can interfere with one another during the training process, and generalize poorly when applied to a *new previously unseen* target domain [283, 207]. This phenomenon is referred to as negative transfer [255].

To address this challenge, we propose an inter-domain similarity-based approach to recommend relevant training data by selecting most similar source domains to a target domain. Conventionally, inter-domain similarity measures compute the discrepancy between the distributions of source domains data and target domain data [74, 44, 214]. However, this is not suitable in our case due to the *unavailability* of target domain data during training. Consequently, we rely on a task-specific deep metric learning based framework which requires solely metadata about the target domain during inference for training data recommendation. Metadata provide contextual description about the target domain, and are modeled as user query. Once training data are recommended, target model is trained. For this, we propose an ensemble learning framework to combine multiple learning algorithms. Ensemble learning generally yields better generalization performance than simply one model. As target task, we focus on predictive modeling using time series data.

We consider the use case of building energy modeling. More particularly, we focus on buildings on which no operational data are available, such as newly built or newly renovated buildings. Metadata about the unseen target building mainly consist of design properties that are easy to access, such as the building's use type (residential, industrial, or commercial), shape, size and age.

The remainder of this chapter is structured as follows. Section 5.2 gives a brief review of most related works. Section 5.3 provides an overview on our proposed methodology. Section 5.4 and Section 5.5 describe the operation of respectively training data recommendation and predictive modeling components. Section 5.6 illustrates the application of our proposed methodology for the use case of building energy modeling, and discusses other possible use cases. Finally, in Section 5.7, we draw conclusions and present suggestions for future research.

5.2 Originality of our Methodology

Proposed approaches addressing the domain shift challenge in multi-source data fall into the broader field of knowledge transfer, and more particularly *multi-source domain adaptation* and *domain generalization* sub-fields (more details in Chapter 4). Domain generalization assumes that the target data are not accessible during training and aims to leverage multiple source domains, as in our work. However, we assume that a contextual description of the target domain (*metadata*) is available to perform source domain selection. We argue that a preliminary selection of most similar domains is necessary when multiple source domains are available, in order to effectively avoid negative transfer [210].

Source domain selection has been investigated in the context of multi-source domain adaptation [74, 49]. It mainly consists in selecting source domains that are the most relevant to the target domain, or assigning a weight to each source domain depending on its similarity to the target domain. However, this generally requires unlabeled data from the target domain to compute similarity measures between target domain and available source domains (see Chapter 4). Instead, we focus on domain selection with *no target data* available during training. Hence, we exploit target domain metadata that are easily accessible. Mancini et al. in [162] propose to use target domain metadata and model domain dependencies using a graph in the context of predictive domain adaptation. Authors in [162] propose to build multiple domain-specific models on each source domain, and then regress a model for the target domain based on nearby domains in the graph. Instead of training multiple domain-specific models, we propose to initially select data from *most relevant* source domains using target domain metadata, and use the data to build a model for the unseen target domain. Metadata about target domain is also modeled as a user query. Moreover, we propose to combine multiple learning algorithms simultaneously. For this, we use an ensemble learning framework. As a result, our framework is by design *generic* and *easily extensible*.

Distance between domains in [162] is computed as a measure between their respective metadata. However, small distance between domains metadata does not necessarily imply a small distance between these domains data, on which final task performance is based. Thus, we use a similarity metric learning approach that captures similarities between domains data, while requiring only domain metadata during test to identify most similar source domains.

Yang et al. in [272, 273] proposed to exploit available metadata about domains or tasks for respectively multi-domain learning and multi-task learning. Metadata in [272] consisted of descriptors that semantically characterize the corresponding domain or task, and that are fed into the model as additional input during training along with data. Instead of combining domain metadata and data, we propose to exploit only target domain metadata for source data selection. Then combine data collected from most similar source domains to train predictive models. This way, we are able to address the domain generalization setting in which no target domain data are accessible during training.

As use case, we consider the application on building energy modeling tasks. Transfer learning is recently being investigated for this task [206, 86, 107]. However, these works focus on domain adaptation, and therefore on historical data scarcity rather than their total absence. To the best of our knowledge, our work is a first cross-building knowledge transfer attempt for predictive modeling of building energy consumption when no historical data are accessible during training.

5.3 Overview of the Proposed Methodology

We present a workflow for relevant existing data recommendation and reuse in the general context of cross-domain predictive modeling tasks. Our system main objective is to train accurate predictive models for new previously unseen target domains based solely on their metadata. As such, we distinguish between two types of information: data and metadata. Domain data are a collection about recorded facts such as measurements, observations or descriptions about anything. On the other hand, domain metadata provides contextual information to the data and the domain itself. For instance, in the framework of building energy modeling, data are produced by the building during its operation, such as measurements of energy consumption and weather conditions. Whereas, metadata summarize contextual information about the building, such as the building type, size, year of construction, location, number of occupants, etc. We should note that various definitions of “metadata” exist depending on the field of study. The definition that we have provided is thus specific to our work and should not be confused with similar terms used in, for example, the field of data management. Figure 5.1

illustrates the relation between domain metadata and domain data in our proposed methodology.

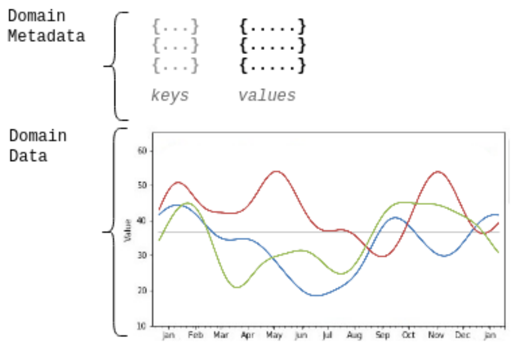


Figure 5.1: Relation between domain metadata and domain data in our proposed methodology. Domain metadata offers a key-value contextual description of the domain. Domain data is historical time-based profiles that we use for predictive modeling.

Our workflow is composed of two main processes. The first process is responsible for recommending most similar source domains with respect to the new previously unseen target domain metadata. The second process is responsible for collecting data from the recommended domains and training a predictive model. We model the input target metadata as a user query. Upon receiving a query, our workflow is launched. We present an overview of our methodology in Figure 5.2. Hereafter, we detail each service of the workflow.

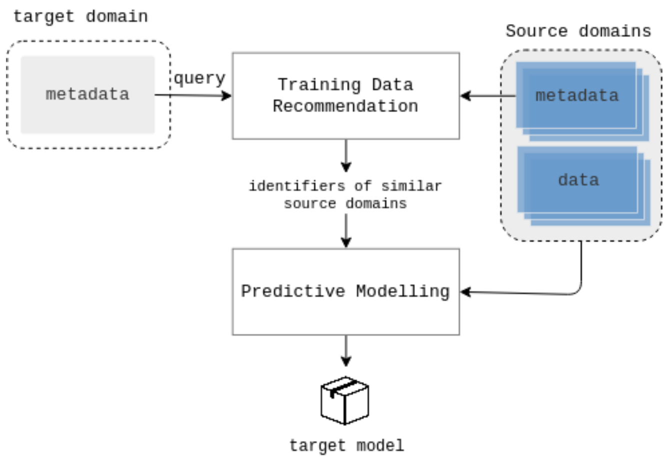


Figure 5.2: Overview of cross-domain knowledge transfer workflow.

1. **Training Data Recommendation** We use a novel inter-domain deep similarity metric learning approach to build a recommendation framework that selects the most similar source domains to a target domain. Inter-domain similarity must rely solely on the target domain metadata and not actual data, given the fact we are interested on the non-availability of historical data. However, predictive modeling depends mainly on such data and therefore recommended source domains

must have similar data patterns as in the target domain. To mitigate this issue, we propose a Siamese network [40] based recommendation framework whose training logic relies simultaneously on metadata and historical data within domains. During inference, it only requires the metadata of the target domain. In brief, our similarity metric learning approach will learn a domain-level feature representation, so that similar domain pairs (having similar data patterns) are mapped close to one another in the feature space, and dissimilar domain pairs are mapped far from one another. This is described in further details in Section 5.4.

2. **Predictive Modeling** Once similar source domains are identified by the training data recommendation component, their corresponding multivariate data are retrieved and combined to form a training dataset for predictive modeling task. Based on the loaded dataset, we learn predictive models from historical data of recommended source domains. These predictive models will later allow us to accurately predict future changes of the new previously unseen target domain described in the user query (metadata). In this work, we use an ensemble learning technique to combine trained predictive models. This establishes the generic nature of our query-adaptive methodology, and ensures better generalization performance as described in Section 5.5

5.4 Training Data Recommendation

Training data recommendation allows to perform a preliminary contextual selection, which is arguably required when generalizing the applicability of cross-domain knowledge transfer. Predictive model training will be subsequently limited to data retrieved from source domains that are sufficiently related and similar to the target domain described in the query. This also avoids negative transfer [210] by filtering out unrelated source domains. Common approaches for domain selection compute the discrepancy between source domains data and target domain data [74, 49]. However, as previously established, we focus on the case of *unavailability of historical data* in the target domain.

The main challenge therefore lies in the definition of an adequate domain-level similarity metric that is simultaneously:

1. defined suitably to reflect similarity between target domain and source domains data patterns. This is due to our target task of predictive modeling, which is mainly based on historical data.
2. not require data about the target domain during inference.

Consequently, we propose a training data recommendation process that requires solely contextual metadata about the target domain in order to select most similar source domains. Metadata will provide a minimal a priori knowledge about the *new previously unseen* target domain, and therefore consist of contextual information that are easy to access. Our recommendation approach relies on *deep metric learning*. Metric learning [134] aims to automatically find a task-specific distance function to measure the similarity between samples. Metric learning techniques may be either supervised or unsupervised. Unsupervised metric learning maps useful information into a low-dimensional subspace. Supervised metric learning formulates the problem as an optimization problem with an objective function on labeled training data [160]. Objective function is designed with respect to the target task.

Another closely related concept to metric learning is *similarity learning*. For instance, metric learning can be seen as a subset of similarity learning. A metric or a distance function $d : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ has to satisfy four axioms:

1. Non-negativity: $d(x, y) \geq 0, \forall x, y \in \mathcal{X}$
2. Identity of indiscernibles: $d(x, y) = 0 \iff x = y, \forall x, y \in \mathcal{X}$
3. Symmetry: $d(x, y) = d(y, x), \forall x, y \in \mathcal{X}$
4. Triangular inequality: $d(x, y) \leq d(x, z) + d(z, y), \forall x, y, z \in \mathcal{X}$

Examples of metric functions are Euclidean and cosine distances.

Existing metric learning approaches often aim to first find a feature transformation to map examples into a new feature space (or embedding space) and then find a discriminative distance metric in this new space [110]. Figure 5.3 illustrates the concept of metric learning. Transformation can be linear or non-linear [134]. Non-linear methods for metric learning are proven effective in addressing the non-linearity problem, which is often encountered in case of real-world data. Non-linear transformation may be implemented using kernel tricks. However, kernel tricks suffer from scalability limitations [160]. As an alternative, deep neural networks are successfully applied. Combination of metric learning and deep learning is referred to as *deep metric learning*.

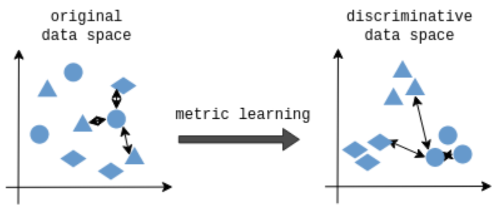


Figure 5.3: Illustration of metric learning by learning a new embedding space that bring similar examples closer to each others and dissimilar examples farther from each other.

In our work, we leverage supervised deep metric learning to learn a distance function between domain pairs to effectively measure their similarity. As such, we utilize labeled source domain pairs to learn a domain-level feature representation (embedding) so that similar domains are mapped close to one another in the transformed feature space, whereas dissimilar domains are mapped as far from one another as possible. This simplifies the following source domain recommendation, by simply selecting the closest domains to the target domain in the new feature space.

We use a Siamese network based recommendation workflow whose training logic relies simultaneously on metadata and historical data within source domains. Figure 5.4 gives an overview of the Siamese network based recommendation framework. Once trained, recommendation workflow is executed at each reception of a query that contains a minimal contextual description (metadata) of the target domain.

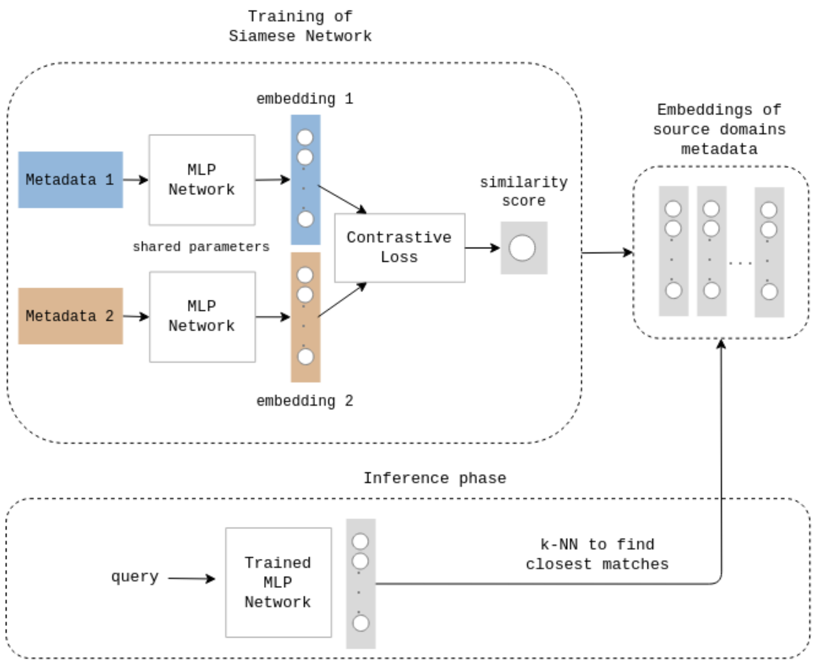


Figure 5.4: Overview on the proposed data recommendation workflow.

The Siamese network [40, 52, 212] is an architecture for non-linear metric learning with pairwise similarity information. As such, a Siamese network learns representations through explicit information about similarity between pairs of objects. Siamese networks were first introduced in [40] in the study of signature verification. It consists of two identical networks both sharing the same weights and architecture. The two networks accept distinct inputs that are then joined via a loss function. The goal is to obtain a representation that preserves the distance between similar entries. This way similar domains can be matched and related through their learned representations.

5.4.1 Deep Metric Learning with Siamese network

Siamese networks have commonly been trained using the contrastive loss function. Contrastive loss function [99] is employed to learn the shared parameters vector W of a parameterized mapping function G_W that ensures that semantically similar examples are embedded close to one another, while semantically dissimilar examples are embedded far from one another. We use the same contrastive loss function defined in [99]. However, in our methodology, we work with continuous (or soft labels) rather than binary labels (or hard labels). Let X_1 and X_2 be the input pair. Let Y the label associated to this pair, Y is close to 0 if X_1 and X_2 are deemed similar, and Y is close to 1 otherwise.

The label Y can therefore be seen as score of similarity between input pair. In our case study, X_1 and X_2 consist in the metadata vectors of respectively a source domain 1 and a source domain 2, whereas Y consists in the scaled distance between historical data of the pair of input source domains into the range of $(0, 1)$.

The parameterized distance function D_W to be learned between inputs X_1 and X_2 is defined as the Euclidean distance between the outputs of mapping function G_W, i.e. $D_W(X_1, X_2) = \|G_W(X_1) - G_W(X_2)\|_2$. To shorten notation, $D_W(X_1, X_2)$ is written as D_W. We use the general form of contrastive loss function $\mathcal{L}(W)$ which is defined as follows [99].

$$\mathcal{L}(W) = \sum_{i=1}^{P} L(W, (Y, X_1, X_2)^i) \tag{5.1}$$

$$L(W, (Y, X_1, X_2)^i) = (1 - Y)L_S(D_W^i) + YL_D(D_W^i) \tag{5.2}$$

where $(Y, X_1, X_2)^i$ is the labeled sample pair of index i, P is the number of training pairs, L_S is the partial loss function for a pair of similar domains, and L_D is the partial loss function of a pair of dissimilar domains. L_S and L_D are designed so that D_W has low values for similar inputs and high values for dissimilar inputs [99]. As such, L_W has low values if similar inputs have "closer" embeddings and dissimilar inputs have "farther" embeddings. L_S and L_D are written as follows :

$$L_S(W, X_1, X_2) = \frac{1}{2}(D_W)2 \tag{5.3}$$

$$L_D(W, X_1, X_2) = \frac{1}{2}max(0, m - D_W)^2 \tag{5.4}$$

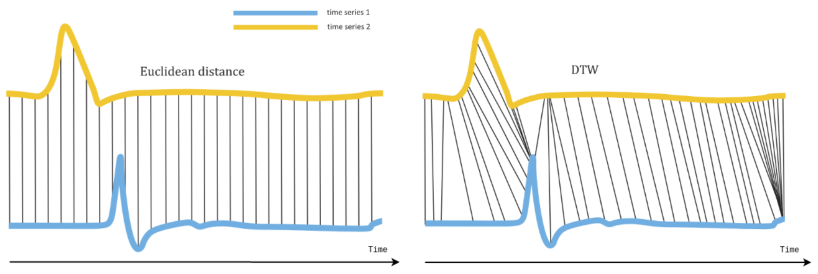
where $m > 0$ is a margin that is used to hold constraint, i.e. when two inputs are dissimilar, and if the distance between them is greater than a margin, they do not contribute to the loss. This ensures that no computations are wasted on enlarging distance between embeddings of dissimilar inputs when these are distant enough. The contrastive term L_D involving dissimilar inputs is crucial in avoiding the loss reaches zero by setting embeddings G_W to a constant [99].

5.4.2 Data labeling with Dynamic Time Warping

Particularly, in our study, we learn domain-level embeddings so that similar source domains are mapped close to each other in the learned feature space, and dissimilar source domains are mapped far from each other. As aforementioned, the Siamese network learns feature representations via a supervised approach with explicit pairwise similarity information. We therefore need to represent our dataset as pairs of metadata vectors, each corresponding to a domain. For each domain pair, there corresponds a similarity score [113] between their respective historical data. In our work, we mainly focus on time series data given our target task of predictive modeling. As such, selection of most similar domains from the metadata embedding space, will result in selection of domains with most similar historical data.

Several distance metrics are proposed for evaluation of similarity between time series data [251], namely Euclidean distance [76], DISSIM [85], dynamic time warping (DTW) [180], longest common subsequence (LCSS) [249], edit sequence on real sequence (EDR) [48], edit distance with real penalty (ERP) [47], threshold queries based similarity search (TQuEST) [12], etc. In our work, we chose to label domain pairs using Dynamic Time Warping (DTW) distance [219]. DTW was first introduced in the context of speech recognition [219], and is a widely accepted distance measure for time series data [124, 67].

Namely, DTW is part of the so-called *elastic measures* for computing similarity between time series data. Elastic measures allow comparison between one-to-many points in the time series. This by contrast to *lock-step* measures which only allow comparison between fixed pairs of points (one-to-one), such as in L_p norms [251]. Figure 5.5 illustrates the difference between Euclidean distance and DTW. Another type of similarity measures is *edit distance based* measures. Edit distance was used in the context of string, by computing the minimum number of operations needed to convert one string into the other. Authors in [251] conducted a comparative study between different distance measures for time series data, and established that DTW generally performs better than lock-step measures (e.g. Euclidean distance, DISSIM) and TQuEST, and very similarly to some edit distance based measures (e.g. LCSS, EDR and ERP). Euclidean distance and other L_p norms are not suitable in the context of time series because of time shifts and time distortions. DTW is on the other hand able to measure similarity between two time series which may vary in time and speed.



(a) Euclidean distance **(b)** Dynamic Time Warping

Figure 5.5: Euclidean distance vs dynamic time warping [202]. DTW allows similar points to match even if they are out of phase in the time axis.

DTW computes similarity by temporally aligning (or *warping*) two time series in an optimal manner by minimizing the distance between them. Suppose we have two time series $\mathbf{x}$ and $\mathbf{y}$, $\mathbf{x} = x_1, x_2, ..., x_M$ of length M and $\mathbf{y} = y_1, y_2, ..., y_N$ of length N. To align these two time series, DTW first constructs an accumulated cost matrix C of size $M \times N$. Accumulated cost matrix C can be defined using Bellman's dynamic programming, by recursively computing [224]:

$$C_{i,j} = f(x_i, y_j) + \min\{C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1}\} \tag{5.5}$$

for $i = 1...M$ and $j = 1...N$. The first element $C_{0,0}$ is initialized to 0, f is the local cost function. For uni-dimensional time series, $f(x_i, y_j)$ is generally defined as the squared distance $(x_i - y_j)^2$. As such, $C_{i,j}$ is the cost at $(i, j)^{\text{th}}$ element and the minimum accumulated cost from the three adjacent elements. The final DTW measure correspond to the total accumulated cost which is written as:

$$d_{DTW}(\mathbf{x}, \mathbf{y}) = C_{M,N} \tag{5.6}$$

One major drawback of DTW computation is the high time and space complexity ($O(n^2)$). This largely limits its applicability [251]. As in many fields, such as building energy modeling, the time series are relatively lengthy and often high-dimensional, the quadratic complexity becomes nontrivial [161]. Therefore, we use FastDTW. FastDTW [220] provides an accurate approximation of DTW that runs in linear time and space complexity ($O(n)$).

5.4.3 Similarity Search

Once Siamese network is trained, we will utilize the learned mapping function G_W to extract embeddings from all source domains descriptions as background process. Source embeddings will be stored in a data store. At each received query, most similar source domains to the described target domain are identified.

Let X_t be the input target domain's metadata. The final recommendation process will be computing distances between the target embedding $G_W(X_t)$ and all source embeddings, such that :

$$D_W(X_t, X_i) = \|G_W(X_t) - G_W(X_i)\|_2 \text{ for } i = 1, 2, ..., S \tag{5.7}$$

where S is the number of source domains. The recommendation process will be then identifying the references of the most similar k domains via a simple k nearest neighbors (k-NN) algorithm. k is an user-defined parameter.

5.5 Predictive Modeling via Ensemble Learning

Time series data retrieved from recommended source domains allow us to train an accurate predictive model for the new previously unseen target domain. Predictive modeling has for objective to forecast future values of time series.

In our work, we developed a generic query-adaptive framework that allows predictive modeling for multiple target domains. Final users are only required to provide as query the contextual description (metadata) about the target domain, and receive as output the corresponding predictive model. As we focus on time series forecasting, we leverage most used and successfully applied learning algorithms proposed in this context, which are support vector machines and artificial neural networks. Support vector machine was applied to regression estimation, and was therefore called support vector regression (SVR). SVR techniques employ kernels in order to capture non-linear patterns often present in time series data. Commonly used kernels are the polynomial kernel and the radial basis function kernel.

On the other hand, ANNs, and deep learning in particular, allows to automatically extract meaningful features from raw data using nonlinear transformations [23]. Within multivariate time series setting, commonly used deep learning techniques [78] are multi-layer Perceptron (MLP), convolutional neural networks (CNN), recurrent neural networks (RNN) and its specific architectures long short-term memory (LSTM) [284, 41] and gated recurrent unit (GRU). A thorough review of machine learning techniques for time series forecasting is done in Chapter 3.

Given the generic nature of our framework, we need to adaptively select most accurate algorithm (and then model) for each invocation, and hence for each unseen target domain. However, simply comparing and selecting the learning algorithm that yields best results for an evaluation subset of target domains, will not necessary work best for all possible cases that our framework might encounter once deployed. This phenomenon is explained by the "*no free lunch*" theorem (NFL) [263]. Essentially, NFL theorem states that all optimization algorithms perform equally good when averaged over all possible problems. This implies that there is no single algorithm that best perform on all possible problems and data sets.

Consequently, we combine several algorithms, and combine their predictions using *ensemble learning* techniques. Rather than finding one hypothesis that best explains the data, ensemble learning consists on building a set of hypotheses, often called an ensemble, and then combine these predictions of hypotheses to produce final predictions on new data points [66]. Ensemble learning typically yields better generalization performance than any single one of the trained models.

The main ensemble learning techniques are *bagging*, *boosting* and *stacked generalization*. In brief, bagging stands for bootstrap aggregating, and mainly aims to reduce variance. It generates multiple sets from the original training set and with the same size, using uniform sampling with replacement. Then, a weak learner is produced on each generated set of samples. Outputs are aggregated by averaging them for regression, and hard or soft voting for classification. By contrast, boosting mainly aims to reduce bias. It consists in a sequence of weak learners, where each model in the sequence is trained with more focus on samples that were "badly" learned by the previous model. For bagging and boosting, the same learning algorithm is used in order to have homogeneous ensemble.

In our work, we use stacked generalization technique [262, 246]. Unlike bagging and boosting approaches, stacked generalization allows to combine heterogeneous models as in our context. It also targets both variance and bias. There are two types of models in a stacked generalization framework: several base models, also called level-0 models, and one meta-model, also called level-1 model. The main idea behind stacked generalization is to use the level-1 model to learn from predictions of level-0 models. In general, a stacked generalization framework can obtain more accurate results compared to the best level-0 model [262]. Figure 5.6 illustrates the concept of stacked generalization. As such, level-0 models can be trained independently from one another in a parallel way.

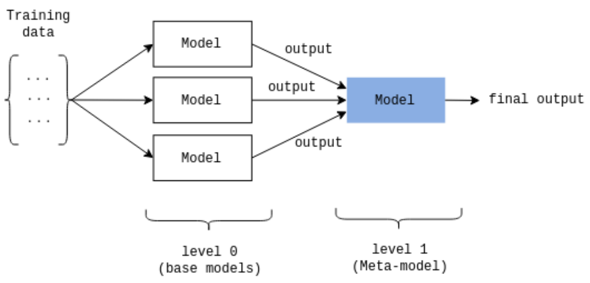


Figure 5.6: Illustration of stacked generalization.

The training data for level-1 model is obtained using cross-validation technique. Given a dataset $\mathcal{D} = \{(y_i, x_i), i = 1..N\}$, where y_i is the target value and x_i represents feature vectors for the i^{th} instance, randomly split the data into K folds $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_K\}$. Define $\mathcal{D}_k$ and $\mathcal{D}^{(-k)} = \mathcal{D} - \mathcal{D}_k$ as respectively the test and the training set of the k^{th} split of the K-fold cross-validation. Given J different level-0 models $\{M_1, M_2, ..., M_J\}$, each M_j is trained on $\mathcal{D}^{(-k)}$ and predicts each sample x_n in $\mathcal{D}_k$. Let z_{jn} denote the prediction of the model M_j on x_n. At the end of all cross-validation process, the dataset assembled from the outputs of J level-0 models is:

$$\mathcal{D}_{CV} = \{(y_i, z_{1n}, z_{2n}, ..., z_{Jn}), n = 1, 2, ..., N\} \tag{5.8}$$

$\mathcal{D}_{CV}$ is the training set for the level-1 model M_{meta}. In our experiments, we use a linear regression model to determine y as a function of $(z_1, z_2, ..., z_J)$. Level-0 models $M_j, j = 1, 2, ..., J$ are on the other hand trained using the dataset $\mathcal{D}$.

The final prediction process will therefore use level-0 models $M_j, j = 1, 2, ..., J$, in conjunction with level-1 model M_{meta}. Given a new target instance, trained level-0 models produce a vector of outputs $\{z_1, z_2, ..., z_J\}$, where z_j is the output of model M_j. This vector is the input to the trained level-1 model M_{meta}, whose output is the final prediction for that instance.

5.6 Use Case of Building Energy Modeling

To verify the effectiveness of our methodology, we perform our experimental study on the use case of energy consumption prediction in buildings. Predictive modeling of energy consumption in buildings is a key task for the optimal management and conservation of building energy within buildings. Yet, accurate predictive models rely heavily on collecting historical operational data about the corresponding building and in a sufficient amount [216, 144, 175].

However, in many cases, historical data are not available for effective training of machine learning models. This is particularly the case in newly built or newly renovated buildings. As such, predictive modeling of energy consumption in renovated buildings can not be based on their historical pattern, given energy efficiency improvements that they had undergone. Moreover, it is common to assess the energy efficiency of buildings before construction or renovation. In such cases, only a contextual description about the future building and its design is available. For all these cases, a contextual description about the target building and its design is available.

Even when few historical data are available, training an accurate model remains a challenge. As for BEM, or more generally time series forecasting, we require training data that cover a long span of time (generally years) to train sufficiently accurate predictive models. For instance, in the case of daily energy consumption prediction, reported research works employ training data for a span of several months (e.g. 10 months) to several years (e.g. 2, 3, 4 years) [8].

BEM therefore represents an interesting use case of our proposed methodology. As such, the training data recommendation allows to recommend relevant training data to a previously unseen target building solely by providing metadata on it. It relies on inter-building similarity metric that assures that similar buildings with similar energy consumption profiles are mapped close to each other in the embedding space, whereas dissimilar buildings are mapped far from each other. Metadata contain minimal contextual information, such as the building type, size, year of construction, location, number of occupants, etc. The deployed framework thus offers a generic method for cross-building predictive modeling. This is particularly convenient given the growing availability of open data in the field of building energy. Existing energy consumption data about multiple source buildings can consequently be obtained. Figure 5.7 illustrates the application of our proposed methodology for the use case of cross-building predictive modeling.

Within the context of predictive modeling of building energy, most applied ML techniques are SVM and ANN. For the experimental evaluation of this use case, we leverage mainly four machine learning techniques; MLP, LSTM-RNN, CNN and kernel SVR. Models are trained to predict daily energy consumption. Building energy consumption depends on multiple exogenous time-based variables, namely weather data

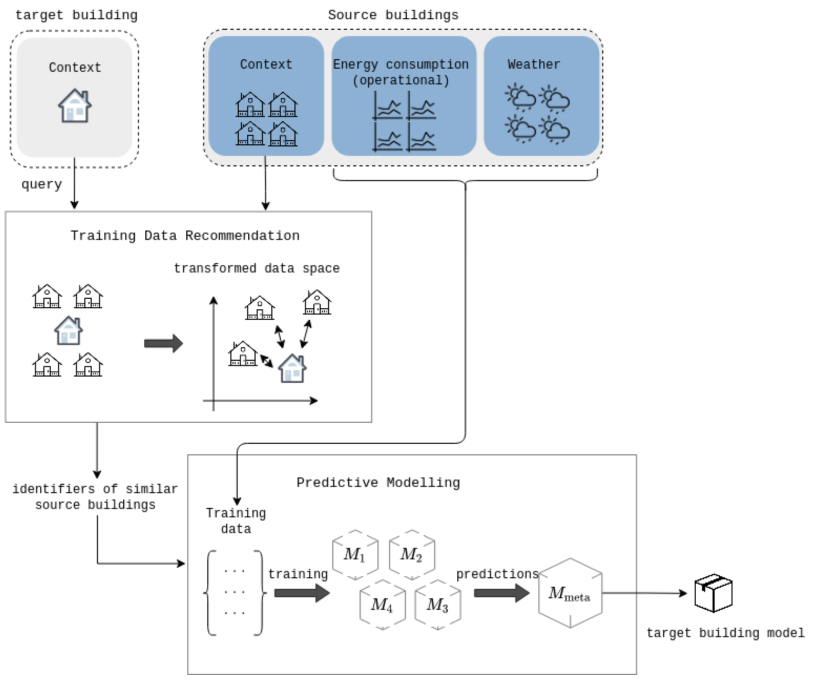


Figure 5.7: Application for the use case of cross-building predictive modeling.

(e.g. air temperature, relative humidity) and calendar data (e.g. weekday, weekend, holiday). Experimental evaluation can be found in Chapter 7.

5.6.1 Genericity of our Methodology

In its design, our methodology performs cross-domain knowledge transfer for a specific task, by using: contextual description about entities (easily acquired metadata), and historical data (with respect to the task), in a way that it requires minimal description and no historical data about a new entity to predictively model it. As such, we believe that our proposed methodology can be considered as a possible solution to a broader range of applications in different fields, and not solely for the use case of cross-building energy modeling.

For instance, time series modeling is increasingly used as a major part of multiple applications, such as healthcare, finances, climate science, biological science, to name a few. Given the data requirements for accurate predictive modeling, we believe that our methodology offers an interesting workaround to effectively perform modeling without the need to have historical data for training and draw advantage from the growth in open data availability. For example, in health care, our methodology may be used for prediction of availability of hospital resources, or patient outcomes, by providing contextual information about the target hospital (e.g. capacity, location). Similarly, in finances, we may want to predict stock prices providing contextual information about a target company.

5.7 Summary and conclusions

Our methodology addresses mainly the issue of non-availability of sufficient historical data in the tasks of predictive modeling. We develop a novel query-adaptive two-step methodology for cross-domain knowledge transfer. As such, two levels of information on buildings are distinguished: (1) metadata which refer to the contextual description about the building, e.g. occupation type, size, location, construction, etc. (2) data which refer to the time series data gathered during building operation.

The first step of the proposed methodology is to recommend most similar source domains in terms of data to a target domain, based solely on its corresponding metadata. For this purpose, an appropriate inter-domain similarity metric is learned using a Siamese network. This ensures the learning of task-specific discriminatory representations of domains. As such, similar domains will be mapped closer together, while dissimilar domains will be mapped farther apart. Consequently, the recommendation process is performed simply by selecting closest learned representations of source domains to the learned representation of the unseen target domain.

The second step is to load historical data from the recommended source domains, and to train multiple heterogeneous predictive models. These models are subsequently combined together via a stacked generalization framework to ensure a robust performance.

Overall, our developed methodology is generic by design as it allows query-adaptive predictive modeling for a diverse range of target domains. To illustrate our methodology, we detail its application for the use case of building energy modeling. To our best knowledge, this work is a first attempt to knowledge transfer across buildings when no operational data available during training. The experimental evaluation of our methodology for the use case of BEM can be found in Chapter 7.

Chapter 6

A Microservice-based Framework for Cross-Building Knowledge Transfer

6.1 Introduction

Our proposed methodology aims to build query-adaptive predictive models for new unseen target buildings based on existing data on multiple source buildings. Hence, we refer to it as *predictive models factory*. The workflow may be logically split into independent components as microservices. Each with a separate concern and requirements. In doing so, the microservice-based architecture is more flexible to changes than a classic monolithic architecture. For the effective software implementation of our predictive models factory, we therefore propose a microservice-based framework. In this chapter, we present the design of our system and detail the flow of data through it.

This chapter is structured as follows. Section 6.2 summarizes the system requirements and the primary actors. Section 6.3 presents our proposed architecture and its main microservices. In Section 6.4, we detail how data is integrated in our system and the used building metadata schema. Section 6.5 concludes this chapter.

6.2 System Requirements

Our system main objective is to train energy predictive models for new previously unseen target buildings based solely on their contextual description. In our application, contextual descriptions concern high-level information about the target building we seek to model, such as its typology, year of construction, location, etc. (details in Chapter 5). Figure 6.1 illustrates the use case of our proposed predictive models factory. As such, predictive models factory is required to integrate and leverage data about multiple source buildings. The source data may be retrieved from open datasets, automatic generation and simulations or real buildings.

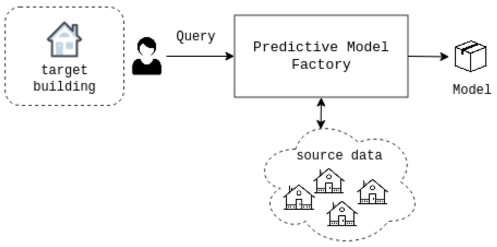


Figure 6.1: Use case of our proposed system.

System actors are mainly building energy professionals and third-party building management systems which seek to accurately model a building on which operational energy data are not available.

6.3 Proposed System Architecture

We establish a microservices-based architecture (MSA) for our predictive model factory. Each individual microservice is fully-independent, self-contained, and specific to a single task. Unlike monolithic applications, the MSA breaks down the application into a suite of flexible, independently deployable and loosely coupled modules that are accessible via a lightweight language-agnostic application programming interface (API). APIs can be based on asynchronous messaging protocols.

MSA offers several benefits, such as an increase in agility in development and delivery, resilience to failure, reliability in operation, maintainability, separation of concerns, and ease of deployment [72, 39]. Compared to service-oriented architecture (SOA), the core intent of the MSA pattern is to limit a service to a single purpose, enabling it to be fully decoupled and thus much more easily scaled and swapped out. Contrary to MSA, component sharing is one of the core tenets of SOA. SOA therefore relies on multiple services to fulfill a business request. Whereas MSA minimizes the need to share components through bounded context, which allows the coupling of a component and its data as a single unit with minimal dependencies. Figure 6.2 shows the various microservices and their coupling in our proposed system. The predictive model factory is composed of the following microservices: building data handling, similarity learning, training data recommendation, and predictive modeling, which are detailed hereafter.

Building Data Handling Microservice Our system is capable of continuously ingesting and integrating data from external providers, such as weather data web services, open energy data hubs and building energy management systems. Time series data about building energy consumption and weather data are respectively stored in the time series store and the weather data store. These two data stores are linked together through the contextual information. In addition, contextual information provides a

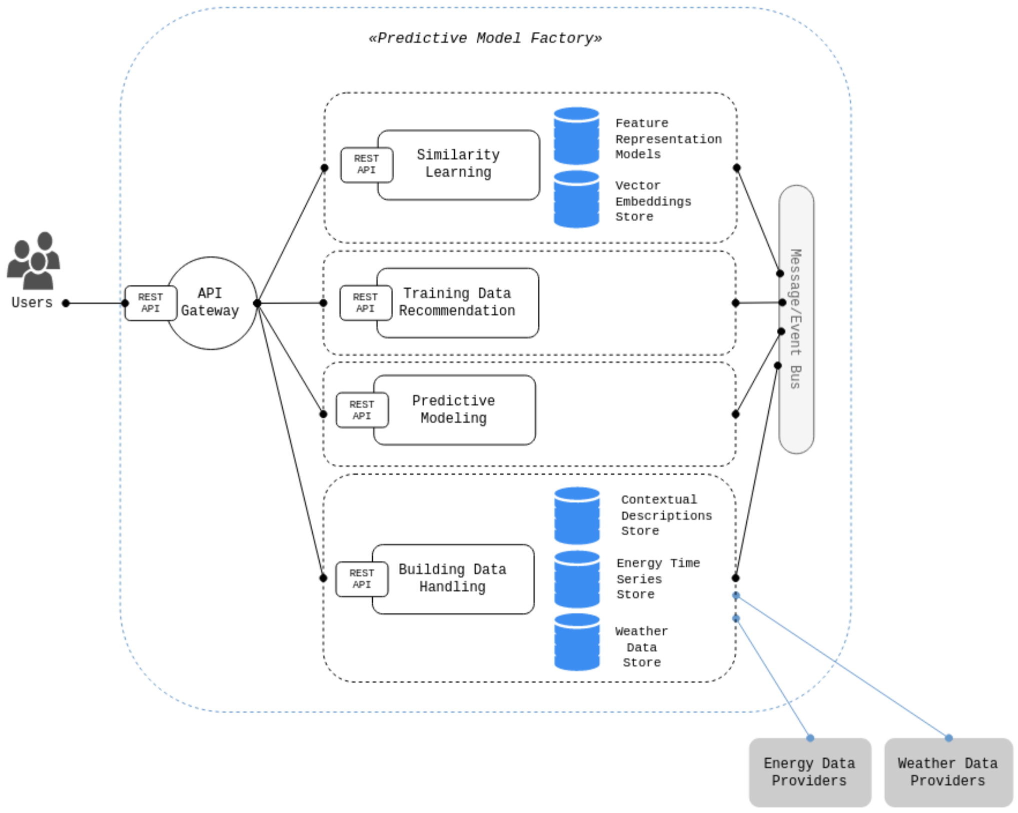


Figure 6.2: The microservice-based architecture for proposed predictive model factory. Dotted rectangles represent individual microservices. Grey rectangles represent external third-party microservices.

high-level description about the building environment, such as layout, age, information about occupants, etc. *Building data handling* microservice plays the role of data provider when learning a relevant similarity metric space, recommending training data and training predictive models. Building data handler provides contextual information and energy consumption time series data to the *similarity learning microservice* to learn an efficient feature representation model. It also provides building-related time series data (about weather and energy consumption) to the *predictive modeling* microservice to learn a accurate predictive model. The actual automation of data integration and data stores population is beyond the scope of our work.

Similarity Learning Microservice *Similarity learning* is designed as a microservice that can run in background regularly in order to perform deep representation learning (described in Chapter 5). Deep representation learning aims to train models that transforms contextual descriptions or metadata about a building to an appropriate vector embedding. Once a new feature representation model is trained, contextual descriptions about available source buildings are loaded to produce corresponding vector embeddings. Vector embeddings of source buildings, as well as trained feature representation model, are consequently stored for later use by other micoservices. We illustrate the relations between contextual description, vector embeddings and feature representation models in Figure 6.3. The workflow of the similarity learning microservice is depicted in Figure 6.4.

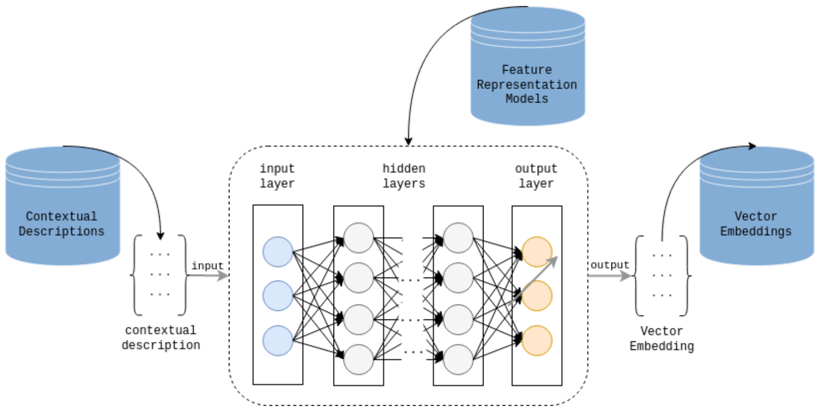


Figure 6.3: Relations between contextual descriptions, vector embeddings and feature representation models.

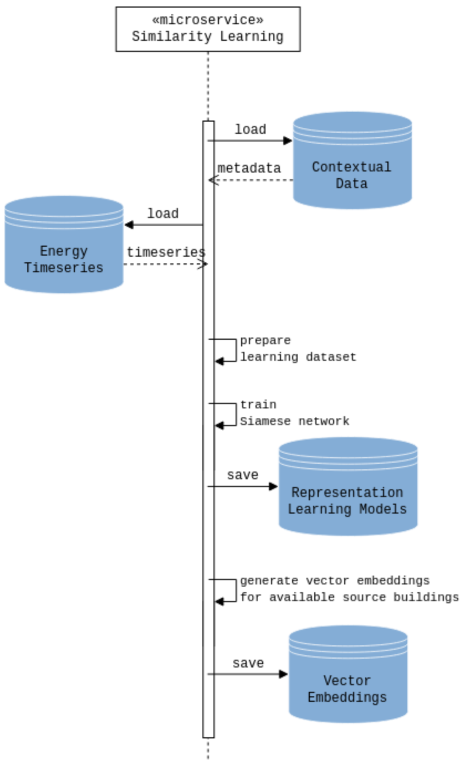


Figure 6.4: Sequence diagram of the similarity learning microservice workflow.

Training Data Recommendation Microservice The entry point of our system workflow is the *training data recommendation* step. Via our system's API, users define the required use case by providing a key-value description of the unseen target building to model via a user friendly interface. No prior knowledge on the target building's energy consumption is needed. The microservice parses the contextual description or the metadata about the target building contained in the user request, and transforms it to an intermediate representation. Vector embeddings are produced using previously trained models by the *similarity learning* microservice (as described in Chapter 5). Most relevant time series data corresponding to most similar source buildings that are already available in our data hub, are then identified and selected. Similar source buildings are identified based on the vector embeddings on the target building and the vector embeddings on other source buildings available within the system. The training data recommendation microservice loads source vector embeddings from their corresponding data store via message queues.

Predictive Modeling Microservice Once similar source buildings identifiers are available, *predictive modeling* service will load corresponding energy data from the time series store and/or weather data store via message queues. Training dataset will be then aligned and prepared using data transformation techniques, e.g. missing data imputation, outlier removal, etc. Finally, predictive model is trained in order to predict future energy consumption for a pre-defined forecasting horizon. In current work, we rely on a recurrent neural network, convolutional neural network and support vector machines for predictive modeling. The final model is then built by combining trained models via ensemble learning techniques. The workflow enabled upon receiving a user request is depicted in Figure 6.5.

Data in our system are shared between microservices following an event-based communication. Microservices therefore communicates via event messages. This enables loose coupling between collaborating microservices and privileges asynchronous behavior. For instance, once similar source buildings are successfully identified by the training data recommendation microservice, their identifiers are shared with the predictive modeling microservice. In addition, in order to have fully independent microservices, we propose to restrict access of relevant microservices to backup copies of databases. For instance, building data handling and predictive modeling microservices share the energy time series store and weather data store, whereas similarity learning and training data recommendation microservices share the contextual descriptions store and the feature representation models store (see Figure 6.5 and Figure 6.4). As it is, database sharing in considered as anti-pattern in a MSA [239], given that it results in significant interdependence and introduces a single point of failure. Alternatively, training data recommendation microservice has access to backup copies of feature representation models store and vector embeddings store. Whereas, predictive modeling microservice has access to an aggregated and reduced time series data from energy data store and weather data store. These backup data stores copies and aggregated data store will be populated periodically through event-based model using publisher/subscriber. This solution is referred to as *data pump model* [183].

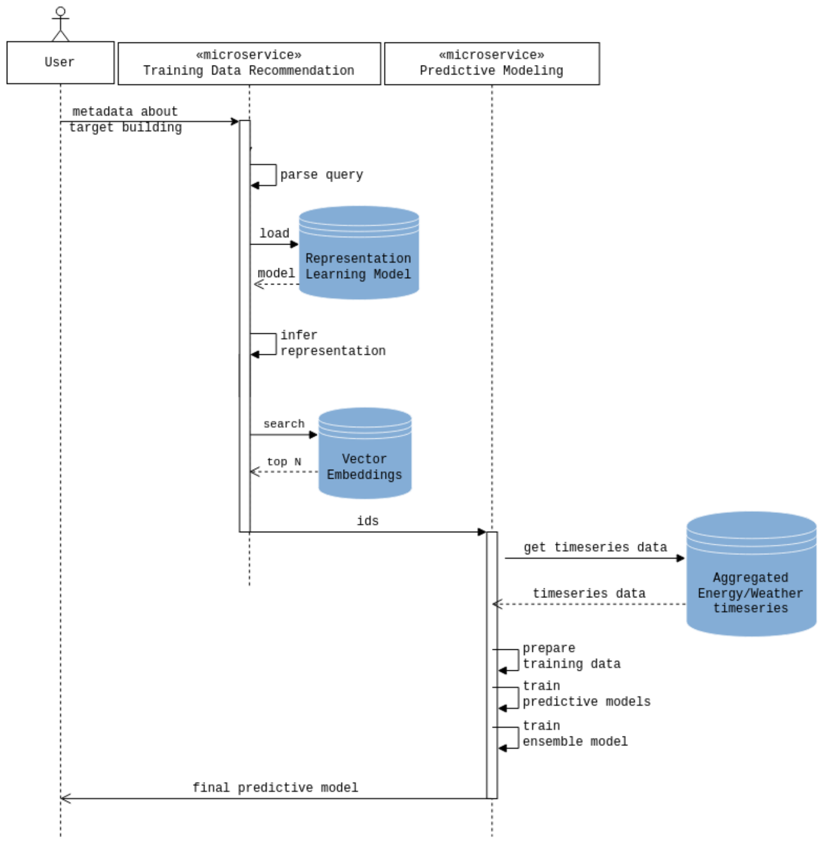


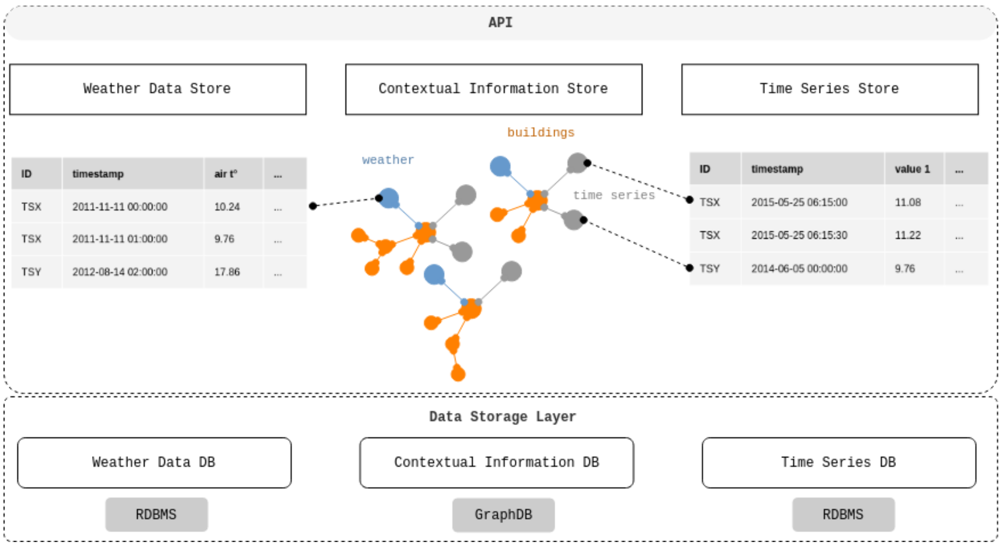
Figure 6.5: Sequence diagram of training data recommendation and predictive modeling microservices workflow.

6.4 Data Specification

To deal with potentially large-scale data, we rely on a multi-modal data store in the back-end. Time series data (weather and energy consumption) are stored in a traditional relational data base management system (RDBMS) like PostgreSQL Timeseries[1]. Our system is transparent to the specific database technology used. Contextual data about buildings and their associated time series is stored in a graph database. As such, we propose an ontology-based data integration for building-related metadata. An overview on building-related data management behind our API is shown in Figure 6.6. Each contextual description about a given source building acts as the core element, by linking both its associated time series data, namely historical energy data and weather data.

Intermediate vector embeddings for the selection of most similar source buildings are stored . Feature representation models trained by the similarity learning microservice are serialized and restored with Python package, Joblib[2]. They can therefore be stored in document-oriented databases, such as MongoDB[3], or databases supporting

[1]https://www.timescale.com/
[2]https://joblib.readthedocs.io/
[3]https://www.mongodb.com/



ID	timestamp	air t°	...
TSX	2011-11-11 00:00:00	10.24	...
TSX	2011-11-11 01:00:00	9.76	...
TSY	2012-08-14 02:00:00	17.86	...

ID	timestamp	value 1	...
TSX	2015-05-25 06:15:00	11.08	...
TSX	2015-05-25 06:15:30	11.22	...
TSY	2014-06-05 00:00:00	9.76	...

Figure 6.6: Contextual information and time series data management component diagram of our proposed system.

BLOB data type like MySQL[4].

6.4.1 Ontology-based Data Integration

As aforementioned, our system allows to integrate data from different data sources, such as open datasets, test buildings or automatic simulations. Multi-source data generally use heterogeneous data formats, such as web pages, text file or XML documents. As result, a different and specifically designed method is employed for each data source to process its data. Alternatively, we propose a unified view of data [143]. This approach for data integration is referred to as *view-based*. The runtime automated data collection and integration processes fall beyond the scope of our work.

For the effective integration of buildings information within our system, we have opted for an ontology-based data integration. The use of ontologies allow better semantic interoperability, automatic verification of data consistency, and flexible querying of data. In brief, an ontology provides *"formal, explicit specification of a shared conceptualization"* [233, 95]. *Conceptualization* refers to an abstract and simplified definition of a phenomenon. *Specification* is the encoding of a conceptualization in a knowledge representation language. As such, an ontology is a representation of knowledge about a domain, as essentially a set of concepts (or *classes*), relationships among these concepts (or *relations*) and instances of these concepts (or *individuals*) [95, 96]. These components constitute a *ontological vocabulary*. A class represents a set of objects within the domain of interest, and is described via its properties (or *attributes*). An attribute has a name and a value of specific data type. An attribute can be a class or an individual. A relation semantically connects classes, individuals, or classes and individuals. Specific types of

[4] `https://www.mysql.com/`

relations exist in an ontology, essentially subsumption and instantiation. Ontologies may also contain restrictions and rules.

Thereby, used metadata ontology in our data hub is required to capture contextual building information from datasets collected from multiple heterogeneous data sources. Each building should also reference its corresponding time series data about weather and energy consumption. Figure 6.7 illustrates the flow of data through our system.

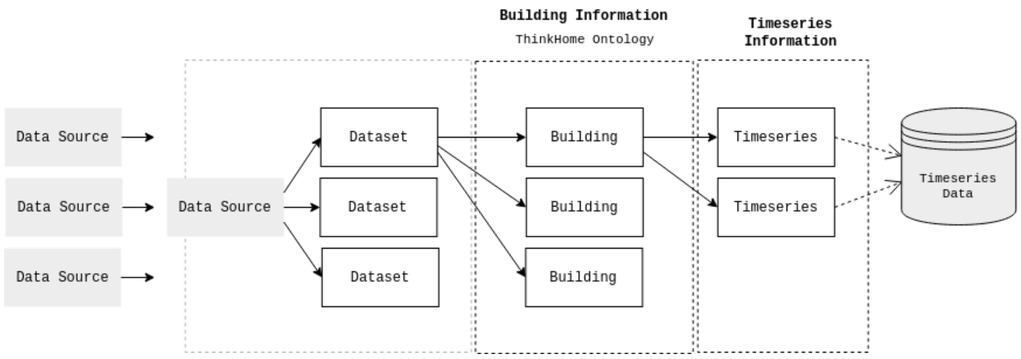


Figure 6.7: An overview of the flow of data through our proposed data hub.

6.4.2 Contextual Building Information

Several ontologies have been proposed in the framework of building information [75, 199], such as BOT [204], SAREF4BLDG [197], ThinkHome [205], DogOnt [31], ifcOWL [193], BOnSAI [232], Brick [15], HTO [101]. The building topology ontology (BOT) [204] is a minimal ontology developed by W3C Linked Building Data Community Group. BOT describes basic topological concepts of a building, such as sites, buildings, storeys and spaces. Hence, it is mainly designed to be reused and extended in order to be adopted for different use cases. The ifcOWL [193] ontology provides an Ontology Web Language (OWL) for Industry Foundation Classes (IFC) model, by replicating the original EXPRESS schema. The resulting graph model allows easy referencing to geographic information systems data, construction material data or product manufacturer data. The ifcOWL therefore covers knowledge related to building design and construction phases of the building life cycle, but not the operation phase. SAREF4BLDG is an extension of Smart Appliances REFerence Ontology (SAREF) [197] ontology for buildings. SAREF4BLDG aims to provide interoperability among IoT devices developed by different smart appliance manufacturers. It is therefore used to decouple information about smart devices from building-specific information. The DogOnt [31] is one of the first ontologies proposed in the context of smart homes. DogOnt formalizes intelligent domotic environment (IDE) elements. It primarily aims to locate smart devices within the building, and describes their capabilities and system interfaces. Hence, it does not provide information about building specific design or energy parameters. For our work, we aim to consider an ontology that capture basic knowledge about building design and energy-related parameters, as well as basic information about building operation (e.g. occupants, setpoints). Hence, aforementioned ontologies do not match our current requirements.

Haystack Tagging Ontology (HTO) [101] is an OWL ontology for the project Haystack. The project Haystack is a domain vocabulary that defines tag taxonomies and entities for mapping building equipment and operational data. In the current version of our system, we handle building-level aggregate energy data. We therefore do not cover knowledge about sensors, smart meters, and operational data collected from them. However, as future work, it would be interesting to extend or adapt our current ontology to such aspects. In the same sense, the Brick [15] ontology proposes to represent buildings and their subsystems. Similarly to the project Haystack, brick uses a tagging scheme to promote flexibility of annotation, while associating tags a semantic model.

The ThinkHome ontology is published by the university of Vienna. The unique characteristic of ThinkHome ontology (with respect to our work) is that it includes all relevant energy related concepts, to perform energy analysis and simulation. Knowledge is logically split into independent modules, as BuildingOntology, ActorOntology, EnergyResourceOntology, etc. For instance, basic information about building design and physics (e.g. layout, materials, walls) are described in the BuildingOntology module, whereas information about actors and their preferences for the smart building system (e.g. age, gender, thermal comfort) are described in the ActorOntology module. The concepts related to smart home automation (e.g. sensor networks) are described in the EnergyResourceOntology module. As such, we have chosen to utilize these Thinkhome metadata schemas to represent contextual information about source buildings. The thinkHome ontology has however some limitations, namely lack of documentation and absence of an ontology license, which may limit its re-usability [75].

ThinkHome ontology [205] In our current version of predictive model factory, we propose to utilize both ThinkHome BuildingOntology and ActorOntology. The BuildingOntology provides basic knowledge about building design and operation, and includes the required concepts for energy-efficient and optimized control processes for smart homes. The ActorOntology provides information about the actors within the building and their preferences. The OWL definition of the ontology is generated using XSLT (eXtensible Stylesheet Language Transformations) from gbXML schema (green building XML)[5]. As a result, BuildingOntology concepts and semantic relations are derived from gXML. The gbXML is the most widely used data model for building energy modeling and simulation systems. In our work, we particularly use gbXML for the generation of our evaluation dataset (see Chapter 7). As such, a building is situated in a campus. The location of the campus is defined as its global origin using latitude, longitude, azimuth orientation, elevation, etc.

[5]https://www.gbxml.org/

The ShellGeometry class is defined as a union of closed shells, where there is no intersection of any two of the given shells. A closed shell is the collection of surfaces bounding a space or a building. Each facet of the closed shell is defined by at least three Cartesian points. These Cartesian points define a three-dimensional plane. All coordinates thus must lie in the same plan. Heat transfer to or from spaces in a building happens through surfaces. A surface can be an exterior wall, a roof, an interior wall, a ceiling, etc. Similarly to gbXML, the BuildingOntology represents surface geometry in two ways: planar and rectangular. Planar geometry uses Cartesian point coordinates which defines the position, shape and area of the surface. Rectangular geometry uses numerical values for height, width, tilt and azimuth. Each surface references a construction. A construction consists of a construction material composite. A construction material composite is composed of material layers, is characterized by solar absorptance, reflectance, an U-value, etc. U-value expresses the thermal transmittance of the composite. Thermal characteristics about each material layer include its thickness, conductivity, density, R-value, etc. R-value measures the thermal resistance of a material to the transfer of heat across it. Openings are modeled as co-planar to surfaces they belong to. An opening can be a door, a window, just air, etc. Characteristics of an opening include U-value, it glazing conductivity, transmittance, etc. We provide an overview of a subset of the concepts of ThinkHome BuildingOntology and ActorOntology in respectively Figure 6.8 and Figure 6.9.

An actor can be human or a system agent. A human actor is characterized by its gender and its age. Each human actor adjusts ranges that provide acceptable indoor environmental conditions for operative temperature (for heating and cooling), humidity, lighting, etc. These preference values can be adjusted differently for each day. The set of preference values associated to hours of a day forms a preference profile.

In the current version of our building metadata schema, we re-use the Thinkhome BuildingOntology and AtorOntology to describe building design and thermal parameters, and basic information about occupants. We therefore start by aligning them via appropriate relations connecting relevant concepts from both ontologies. Alignment is performed with subsumption and equivalence relations. Namely, actor concept and building concept were matched via subsumption relation.

We also include additional properties to denote supplementary information required for our application. We present added properties in Table 6.1.

6.4.3 Time series Information

Each source building in our data hub is generally associated to time series data about outdoor weather and energy consumption. We model time series information as shown in Figure 6.10. A timer series has an identifier, a description, and is associated to a building, referenced by its identifier or URI (unified resource identifier). A time series instance also references the identifier or primary key of the data in the corresponding database. It contains one or many observation or variable sets. An observation set is

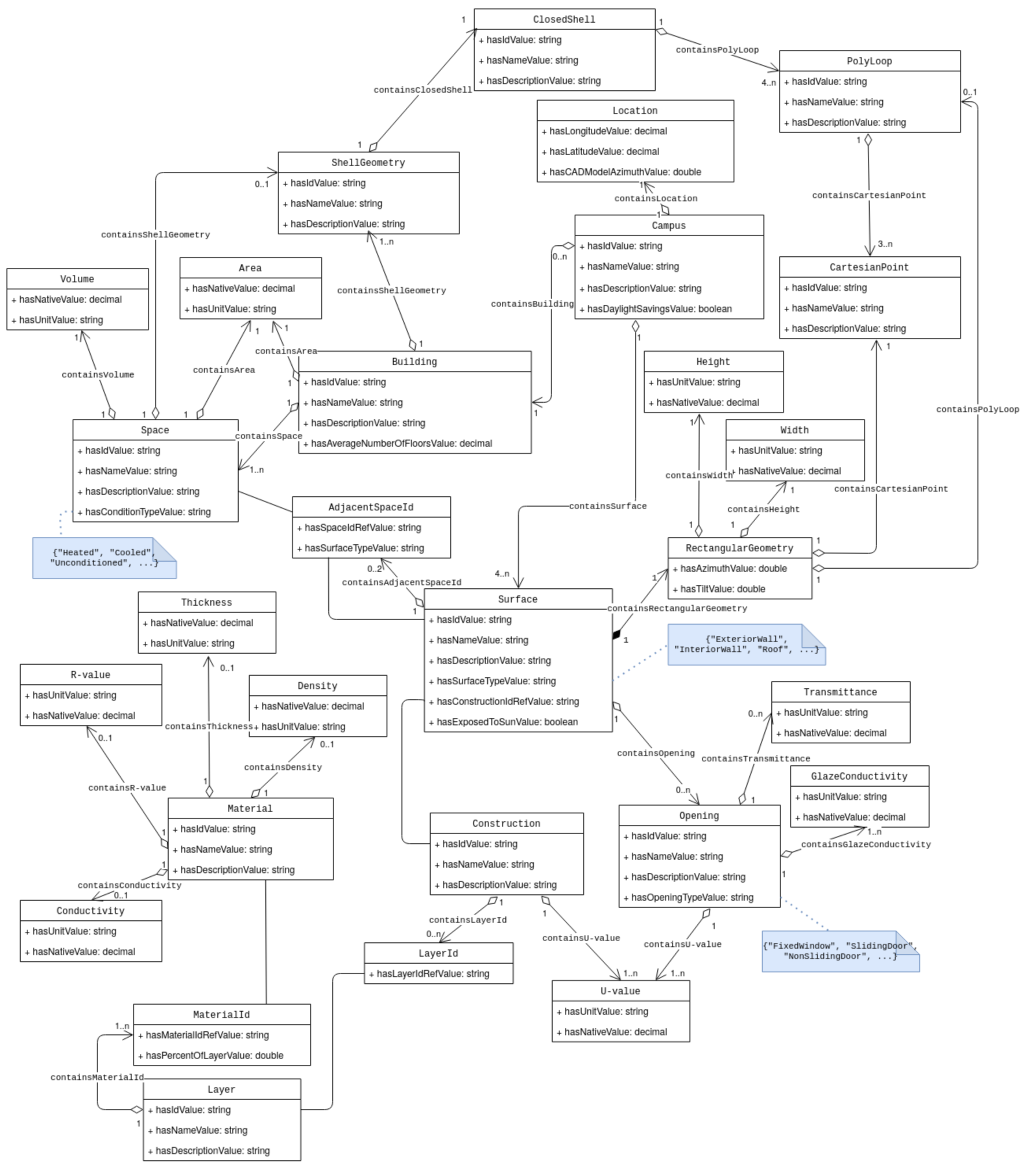


Figure 6.8: Class diagram of a subset of ThinkHome BuildingOntology concepts and properties.

characterized by a name, an unit value, and is composed of one or many observations or measurements. The observation are values taken at different time steps.

6.4.4 Example of Instantiation

The described metadata schema was instantiated as an example using the REFIT Electrical Load Measurements dataset [181]. The dataset contains pre-processed electrical consumption measurements for 20 households at aggregate and appliance level. Descriptions about each building comprises information related to occupancy (number, age, gender, etc.), size in terms of number of bedrooms, construction year, typology,

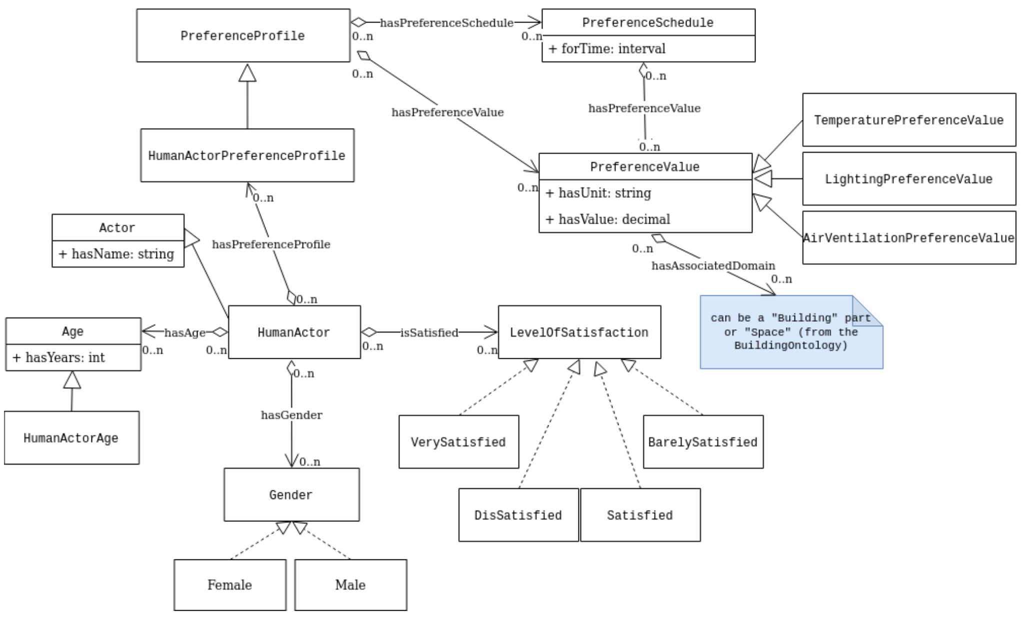


Figure 6.9: Class diagram of a subset of ThinkHome ActorOntology concepts and properties.

Table 6.1: Additional properties in the used building metadata schema.

Property	Domain	Description
ASHRAE climate zone	Building	Climate designation used by the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE). ASHRAE climate zone depends on building location. In total, there are 16 climate zones: 1A (very hot, humid), 2A (hot, humid), 2B (hot, dry), 3A (warm, humid), etc.
Built form typology	Building	Type of the built form. We distinguish between; detached, attached from one side (or semi-detached), attached from two sides (or terraced), and attached from three sides (or apartment).
Volume	Building	Measure of the building volume. It re-uses the *Volume* concept associated to *Space*.
Construction year	Building	The year of construction of the building will provide information about its age, which is a strong indicator of energy consumption [5]. For this object property, we propose two ways: the exact year or an interval between two years.
Presence at building	HumanActor	We provide information about whether an actor is permanently present at the building or not.

and total number of owned appliances. All buildings are located near to the town of Loughborough, UK. Figure 6.11 shows the example of an instantiation as stored in

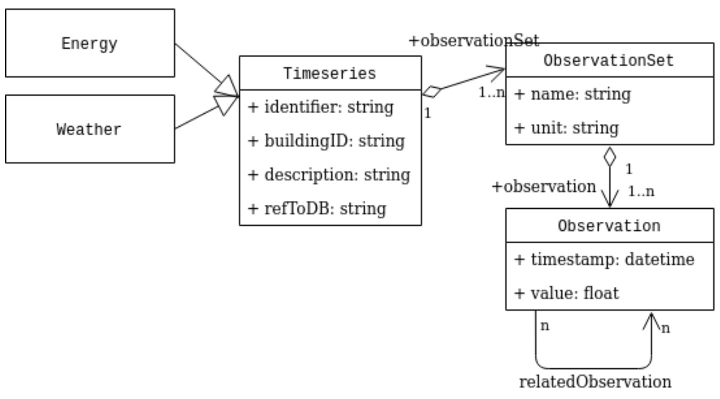


Figure 6.10: Class diagram of Time series related data model.

GraphDB[6]. For reading contextual information contained in the metadata model (instance of metadata schema), the similarity learning microservice uses SparQL[7] query language.

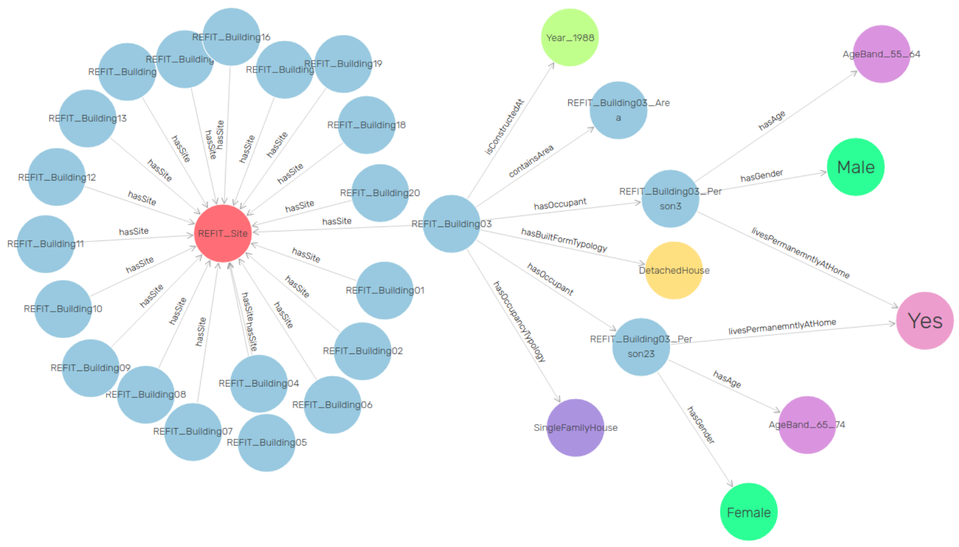


Figure 6.11: An example of an instantiation of our ontology-based unified view of building metadata using REFIT dataset.

6.5 Summary and conclusions

To summarize, we present a microservice-oriented architecture for the implementation of our proposed predictive models factory [138]. Logically independent workflows are therefore modeled as microservices with single purposes and separate data sources. Thereby, four microservices are developed, namely building data handling, similarity

[6] https://graphdb.ontotext.com/
[7] https://www.w3.org/TR/rdf-sparql-query/

learning, training data recommendation and predictive modeling. Building data handling microservice is responsible for ingesting and integration building energy data and metadata within our system. Similarity learning microservice builds task-specific feature representation models from source buildings metadata and based on similarity information between corresponding data. Training data recommendation microservice select most similar source buildings at each reception of an user query describing the target building. Historical data about selected source buildings will serve as training for target predict models. Predictive modeling microsevice builds target predictive models based on recommended training data.

By design, MSA offers increased evolvability and scalability of the system as well as accelerated development velocity. Our designed system also performs integration of building metadata and time series data collected from multiple sources into a unified global view. In this view, building metadata models are transformed into ontologies. Ontologies provide a common semantic framework to represent domain knowledge. Time series data associated to each building are referenced from the relevant ontology concepts. We extend for this purpose ThinkHome BuildingOntology and OccupantOntology.

Chapter 7

Experimental Evaluations

7.1 Introduction

The present experimental study considers the use case of building energy modeling, which aims to accurately predict future energy consumption in buildings. Energy consumption prediction is important for the efficient planning and operation of power systems. Our methodology transfers knowledge from several source buildings, to one new, previously unseen target building. A preliminary training data recommendation is performed by selecting most similar source buildings to a target building using only contextual features about respectively available source buildings and the target building.

The remainder of this chapter is structured as follows. In Section 7.2, we detail our evaluation data set and the data generation process. Section 7.4 details the experimental setting and results of the evaluation of similarity learning and training data recommendation processes. Section 7.5 details the experimental setup and discusses experimental findings in the evaluation of predictive modeling component. Lastly, in Section 5.7, we summarize and conclude the chapter with future directions.

7.2 Energy Consumption Data Generation

For the efficient execution of the developed predictive model factory, two levels of information about buildings are leveraged, namely metadata and data. Metadata consist of a minimal contextual description about the building that is easy to acquire in real world scenarios, namely information about its design, year of construction, etc.

Moreover, metadata/data about a relatively large number of different source buildings needed to be available to feed the developed data hub, and to inter-building similarity learning. Review of open datasets that fulfill the above listed requirements yielded the REFIT dataset [181]. However, due to the limited data (20 buildings), the different processes of the predictive model factory could not be decisively evaluated. This was also due to the contextual similarity between buildings in REFIT, namely all available buildings were residential and located in the same city.

Hence, we resorted to the generation of a sufficiently large and heterogeneous evaluation dataset. This dataset features 105 synthetic buildings with different characteristics. The data set is built as follows. Initially, 21 base building models are available. Each building's model was re-parameterized to augment data using a random combination of design parameter values. Randomly generated parameter values follow an uniform distribution. On average, 4 augmented building models were obtained from each base building model. The measurements were later obtained by automatic simulation of building energy models. The 21 base buildings were held out to serve as test set for both the recommendation component and the predictive modeling component. The remaining 84 buildings, obtained after augmentation, served as training set. Note that the validation set and hyperparameter tuning are made from the training set.

Buildings were modeled and simulated using DesignBuilder; a building energy simulation software that uses EnergyPlus as simulation engine [59]. Fig. 7.1 depicts the 3D representations of a subset of seven base buildings. Building models properties provide static descriptions about buildings, and comprise information related to occupancy, total area, volume, building's activity sector and typology, location, orientation, etc. As this work aims to select similar buildings for the predictive modeling task of building energy consumption, some information about the building thermal envelope was included in contextual descriptions about buildings. Namely, thermal conductivity and heat capacity of building external walls, window-to-total-wall-area ratio, and heating and cooling thermostat set points. In addition, we further characterize buildings by their power generation units.

After simulation, energy consumption and energy generation measurements of one year were recorded. This data cover several aspects, including total energy consumption, heating, air conditioning, and domestic hot water generation. In addition, per-site weather data were recorded, mainly outside air temperature, atmospheric pressure, wind speed, wind direction, etc. In this work, we are mainly interested in heating energy consumption and several weather variables. We work with one-day resolution data which were obtained by summing the original data. Input data were also scaled between 0 and 1.

We should note that we have made sure that generated buildings metadata, and hence data, are sufficiently heterogeneous to reduce the risk of bias in our experimental evaluations. This is mainly ensured by random generation of parameter set in our buildings models augmentation process. Heterogeneity of our data can be seen by plotting the distribution of pairwise dynamic time warping (DTW) distances between buildings total heating energy consumption as in Figure 7.2.

7.3 Hardware and Software Setup

The prototypes of the main processes of the methodology are built on a portable personal computer with Intel i7 processor with 8 cores up to 1.9 GHz and 32GB of memory.

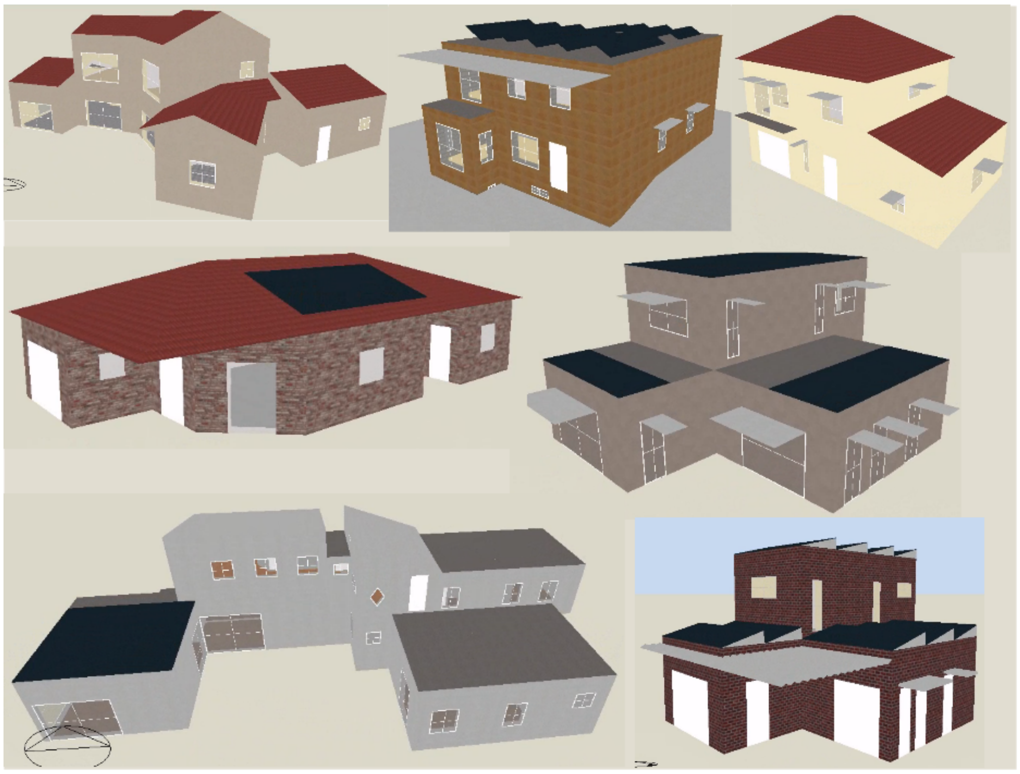


Figure 7.1: 3D view of a subset of base buildings used for data generation.

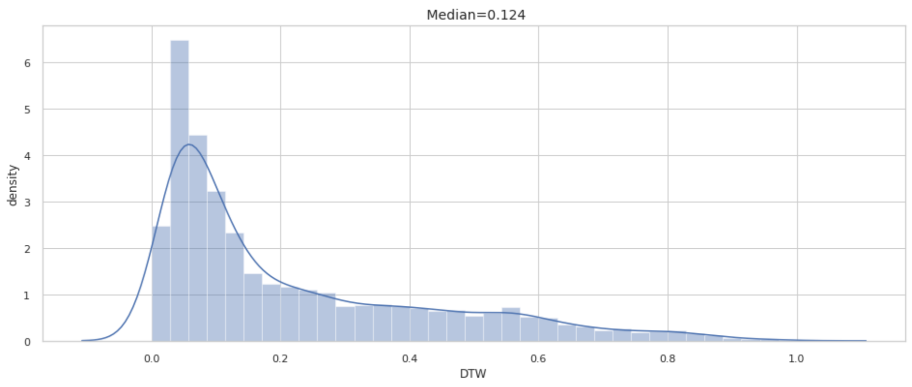


Figure 7.2: Distribution of pairwise DTW distances between available source buildings pairs.

Deep learning models are implemented using the Keras library [51] with Tensorflow backend [169]. Data pre-processing and support vector regression models are implemented using Scikit-learn [194].

7.4 Recommendation of Similar Buildings

As further described in Chapter 5, the workflow is mainly composed of two main processes in the use case of building energy modeling (BEM): (1) Recommendation of training data which relies on deep similarity learning logic to identify most similar buildings, (2) predictive modeling of building energy. Deep similarity learning is performed using a Siamese network that is composed of two identical neural networks sharing the same parameters. These networks are trained to find a feature transformation that maps metadata about buildings into a new feature space, in which new feature representations of similar buildings are mapped close to each others, and new feature representations of dissimilar buildings are mapped far from each others.

7.4.1 Data Preparation

Available metadata about buildings in our generated dataset include features related to building activity's sector and typology, location, number of occupants, and thermo-physical properties of exterior wall materials. From which, we consider a total of 13 features in our training data recommendation workflow. Table 7.1 summarizes the used metadata features.

Input metadata features are scaled to the range of (0, 1). We use the min-max normalization for numerical features. Categorical features (i.e. climate zone, typology) are one-hot encoded. The one-hot encoding transforms each element from original vector into a new vector with n (binary) elements, depending on the number of categories (unique values of the feature) present in the categorical feature. As such, only the corresponding category is set to 1 while the rest of new elements are zeroes. The main advantage of one-hot encoding is that it does not allow machine learning models to assume ordering between different categories.

Table 7.1: Contextual energy features of building metadata.

Feature	Unit	Description
ASHRAE climate zone	-	climate designation used by the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE). ASHRAE climate zone depends on building location. In total, there are 16 climate zones.
Orientation	$^\circ$	angle in degrees ($^\circ$) of the building plan view with respect to anti-zenithal direction, North being 0°. For instance, a value of 45° means North East.
Typology of building activity	-	type of the building occupancy. Different types are modeled as taxonomic hierarchy. We mainly distinguish between residential, industrial, commercial, office, etc.

Number of zones	-	total number of thermal zones in a building. A thermal zone is a space or collection of connected spaces within the building that have the same thermal behavior.
Building heated/cooled floor area	m^2	area of heated or cooled floors. -
Building volume	m^3	-
U-value of external walls structure	W/m^2K	also called overall heat transfer coefficient or thermal transmittance. U-value of an external wall is defined as the rate of heat transmitted through a square meter of the wall with respect to temperature difference between the inside and the outside. A lower U-value thus reflects good thermal insulation properties.
Thermal capacity of external walls structure	kJ/K	also called thermal mass or thermal capacitance. Thermal capacity of an object is defined by the amount of heat required to change its temperature by one unit. It thus reflects the capacity to absorb, store and release heat.
Building area-weighted average U-value (including bridging)	W/m^2K	combines different U-values of various surfaces of the building into a single value. Thermal bridging refers to building's areas that has a greater heat transfer rate than the surrounding materials. These bridges mainly occur at junctions between building elements and insulation materials, or at junctions of building enclosure components, such as wall to window or wall to roof.
Window-to-total-Wall-area ratio	%	ratio of vertical fenestration area to total exterior wall area. The fenestration area is the total window rough opening, including fenestration and frame components.
Number of occupants	-	total number of occupants within a building.

Temperature set-points for Heating and Cooling	°C	expresses the ideal temperature (i.e. the setting of the heating or cooling thermostat) in the space when heating, or respectively cooling, is required.

A subset of these properties is depicted in Figure 7.3

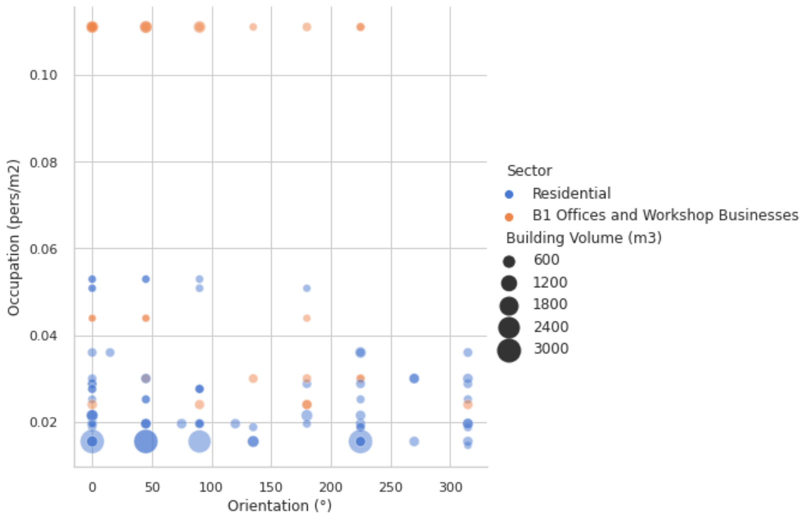


Figure 7.3: Overview on a subset of the dataset characteristics using scatter plot. Namely, building's activity sector, total area (m^2), occupation density (pers/m^2), and orientation (°).

7.4.2 Model Training

We employ a Siamese multi-layer perceptron. Each MLP model is composed of two hidden layers both of size 128. A dropout rate of 0.1 was applied to each of the hidden layer. Dropout regularization [231] in the context of deep learning models consists in randomly ignoring, or "dropping out", neurons (and their connections) during the training phase. As such, a dropout rate of 0.1 means that we are randomly setting 10% of neurons of a given layer to 0 at each update (i.e. 90% probability of retaining any given neuron). This prevents any neuron from co-adaptation on the training data, which otherwise leads to over-fitting.

The Rectified Linear Unit (ReLU) is used as the non-linear activation function for hidden layers. The output layer consists of a fully-connected layer of size 25. Note that model hyperparameters were chosen empirically during validation.

7.4.3 Experimental Results

Several metrics may be used for the evaluation of the training data recommendation component. Namely, we compare between the average DTW distance within the recommended list of source buildings operational data and the average DTW distance between the actual list of most similar source buildings operational data. We refer to this value as *intra-list distance*. The actual list (ground truth) is identified by selecting source buildings with lowest pairwise DTW distances between their data and the target building's data. The predicted list is identified by selecting the source buildings with closest metadata embeddings in the learned new feature space, to the unseen target building's metadata embedding. As detailed in Chapter 5, a metadata embedding of a given building is the output of the trained Siamese network's MLP when providing the metadata of this building as input.

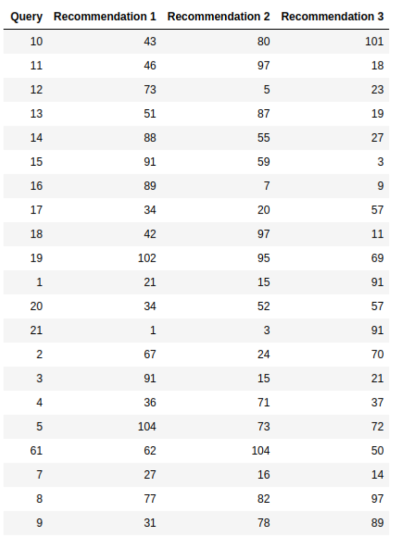
Our predicted intra-list distance is of 42.95 Wh as opposed to a true intra-list distance of 24.79 Wh. Figure 7.4 compares between elements of the actual list and the predicted list of most similar source buildings, when setting the number of recommendations to 3. The number of recommendations refer to the number of source buildings that are selected for each target building. We set it to 3 as an example. As such, if each source building had one year energy-related data (as in our case), predictive modeling would be based on a total of three years data. This is particularly common for daily modeling tasks [8]. In Figure 7.4, we present the identifiers of the three recommended source buildings for each query. Each query corresponds to an unseen target building (from the held-out test buildings), which is also presented by its identifier.

Moreover, we experimentally evaluate the motivation behind our proposed similarity learning framework for the task of selection of most similar source buildings. As such, this raises the question of why not simply select the source buildings that have closest metadata vectors to the metadata data vector of the target buildings, and why would we require to go through a similarity learning framework. For this, we compute the average intra-list distance of metadata vector-based recommendations, and find a value of 163.91 Wh. The proposed deep similarity framework is therefore experimentally proven to be effective, and yields closer recommendations to ground truth.

7.4.4 Discussion

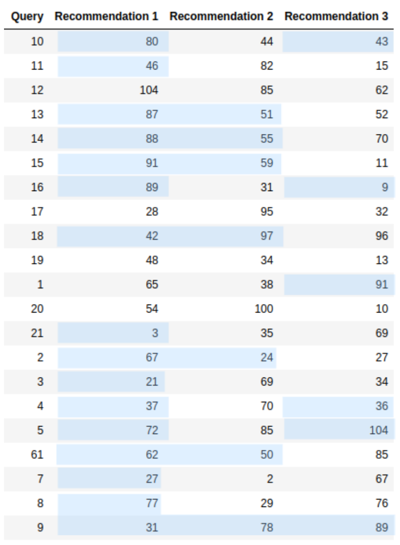
From experimental results, we can notice that using a deep similarity learning framework to learn task-specific metadata embeddings that capture similarity between buildings, yields good performance.

This is particularly useful in the context of a generic query-adaptive predictive model factory, as in this case. For instance, the query-adaptive model factory is required to train predictive models on data retrieved from recommended buildings. Predictive modeling results therefore rely on the quality of source buildings recommendation. By dynamically learning task-specific metadata embeddings, we are able to select most



Query	Recommendation 1	Recommendation 2	Recommendation 3
10	43	80	101
11	46	97	18
12	73	5	23
13	51	87	19
14	88	55	27
15	91	59	3
16	89	7	9
17	34	20	57
18	42	97	11
19	102	95	69
1	21	15	91
20	34	52	57
21	1	3	91
2	67	24	70
3	91	15	21
4	36	71	37
5	104	73	72
61	62	104	50
7	27	16	14
8	77	82	97
9	31	78	89

(a) Ground truth



Query	Recommendation 1	Recommendation 2	Recommendation 3
10	80	44	43
11	46	82	15
12	104	85	62
13	87	51	52
14	88	55	70
15	91	59	11
16	89	31	9
17	28	95	32
18	42	97	96
19	48	34	13
1	65	38	91
20	54	100	10
21	3	35	69
2	67	24	27
3	21	69	34
4	37	70	36
5	72	85	104
61	62	50	85
7	27	2	67
8	77	29	76
9	31	78	89

(b) Prediction

Figure 7.4: Comparison between the actual list (left) and the predicted list (right) of most similar buildings identifiers. Queries stand for the unseen target building and is presented by their identifiers. Common elements between the actual list and the predicted list are highlighted in blue in the predicted list. Order of recommendations is ignored given that we will later combine data from recommended buildings with no regard to it.

similar source buildings using raw building metadata. This avoids the need to thoroughly study metadata features importance, and identify their different weights for the target task. As such, not all features are equally important and contain information about energy consumption (target task).

7.5 Predictive Modeling of Energy Consumption

Once most similar source buildings are recommended, their corresponding operational data are retrieved for predictive modeling of new previously unseen target building. Predictive modeling in buildings aim to accurately forecast future energy consumption, which supports projects at building-level, such as energy-efficient design, benchmarking, or control. Predictive modeling is a complex problem that requires consideration to various variables, mainly historical data, weather conditions, occupants' behaviors, calendar effects, etc. In our work, we mainly focus on historical energy consumption, weather conditions and some calendar effects.

7.5.1 Data Preparation

In our experimental work, the goal of the target predictive model is to predict future daily heating energy consumption after a pre-defined time horizon. Input data consist in related information about historical heating energy consumption, outside air temperature, solar irradiation, atmospheric pressure and weekday/weekend index.

As historical energy consumption data, we use a series of measurements at previous time steps. These measurements are generally referred to as *lag observations* in the context of auto-regressive models. An auto-regressive model is when a value from a time series is regressed on previous values from that same time series. The number of lag observations is generally chosen depending on auto-correlation analysis. For instance, partial auto-correlation function (PACF) can be used to study partial auto-correlation between a time series and its own lagged observations. Figure 7.5 illustrates the partial auto-correlation of heating energy consumption time series for a given building, along with the approximate 95% confidence intervals. We can notice partial auto-correlation coefficients are outside the confidence intervals at lags of 1, 2, 13 and 14. Auto-correlation at these lags is therefore statistically significant. Other buildings available in our dataset showcase similar results at these lags. Given these results, we consider in our predictive modeling task historical energy consumption data for previous two weeks ($[t - 14 : t - 1]$) in order to predict future energy consumption ($[t + 1 : t + h]$, h being the prediction horizon.

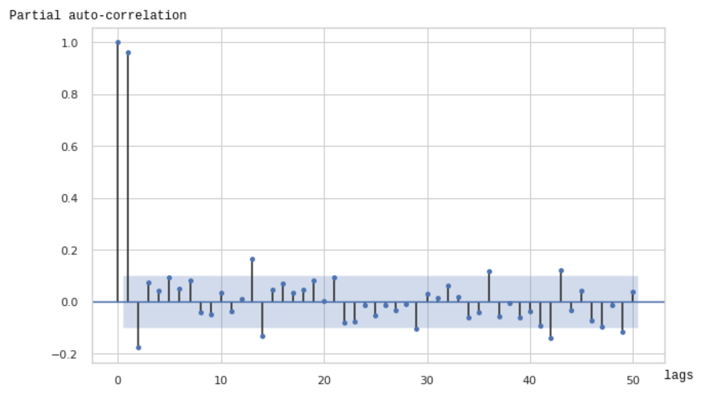


Figure 7.5: Partial auto-correlation plot for 50 lags of heating energy consumption time series. The blue shaded area stands for the approximate 95% confidence intervals. Autocorrelation outside these confidence intervals is a statistically significant correlation while those which are inside the confidence intervals are due to random noise.

For weather variables selection, we consecutively eliminate variables that are not correlated with the target variable (heating energy consumption), and variables that are redundant via a multicollinearity analysis. The choice to work with outside air temperature, solar irradiance and atmospheric pressure was made based on Pearson correlation coefficients between available weather variables and heating energy consumption for several buildings in our dataset. We summarize the matrix of correlation coefficients in Figure 7.6. Dry-bulb temperature is what we usually refer to as air temperature. It is called “dry-bulb” to indicate that the thermometer is shielded from moisture of the air (and radiation). Dew-point temperature is the temperature at which air is saturated with water vapor, and can no longer hold it. As such, if dew-point temperature is close to air temperature, the relative humidity is high. Otherwise, if it is significantly below air temperature, the relative humidity is low.

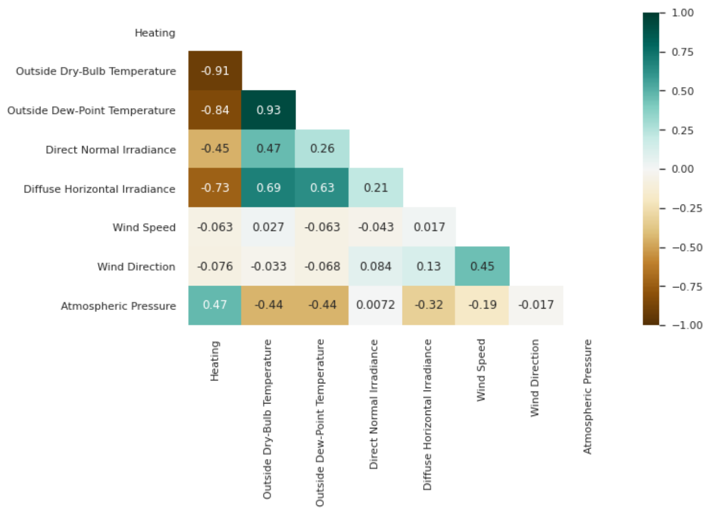


Figure 7.6: Matrix of Pearson correlation coefficients between available weather variables and heating energy consumption variable.

Throughout the day, solar radiation strikes the surface at different angles, ranging from 0° (above horizon) to 90° (zenith). This radiant energy is measured as the solar irradiance. Figure 7.7 illustrates the different components of the solar radiation reaching the ground level. As such, during its travel, solar radiation is depleted by scattering, reflection and absorption by constituents of the atmosphere (e.g. aerosols, clouds, poltants). Direct normal irradiance (DNI) is the component that comes in a straight line from the direction of the sun. Diffuse horizontal irradiance (DHI) is the component of the remaining radiation (scattered or reflected) that is reflected back to the surface [243]. The combination of DNI, DHI and ground reflected radiation (albedo) forms global horizontal irradiance (GHI).

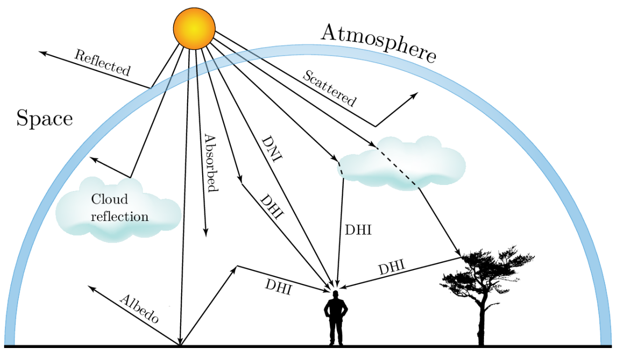


Figure 7.7: Different components of the sunlight (GHI, DNI, DHI). GHI is the combination of DNI, DHI and Albedo [243].

From Figure 7.6, wind speed and wind direction have minimal correlation with heating energy consumption (target variable), hence discarded. We can however notice that energy consumption for heating has very high negative correlation with outside dry-bulb temperature, outside dew-point temperature and diffuse horizontal irradiance. We also notice a relatively high positive correlation with atmospheric pressure, and a relatively high negative correlation with direct normal irradiance (also called solar radiance). Correlation matrix shows that multicollinearity is present in weather data. Multicollinearity occurs when an independent variable can be predicted by other independent variables in a regression model. In our experimental work, we used variable inflation factors (VIF) to detect it. In brief, VIF technique incrementally selects each explanatory variable and regress it against every other variable. VIF therefore measure how well each variable can be explained by other variables. Generally, a VIF above 10 indicates that there is a significant multicollinearity. Following VIF-based multicollinearity analysis, we exclude outside dry-bulb temperature from our predictive modeling dataset. The correlation between heating energy consumption and selected weather variables is further depicted by plotting in Figure 7.8.

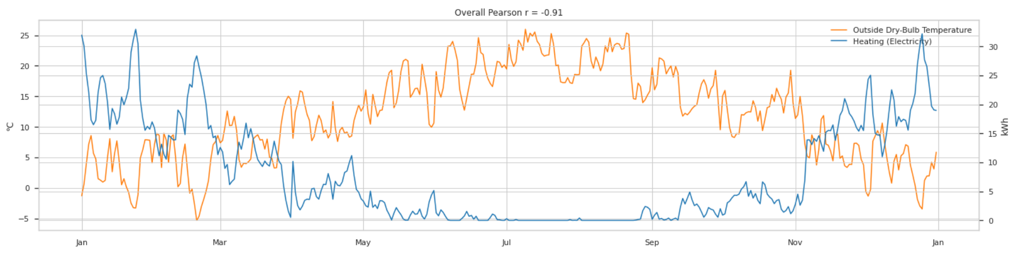
Note that the above-described feature selection is not necessary when working with deep learning based predictive models. However, it helps reduce models complexity.

The target value is a real value or vector denoting the heating energy consumption at future time steps after a pre-defined time horizon. In our experiments, we consider both cases of one-step ahead, and multi-step ahead time series forecasting. For one-step ahead forecasting, our models predict the next day's heating energy consumption as output. For multi-step ahead forecasting, our models predict next week's daily heating energy consumption. To re-frame time series data as supervised learning problem, we use sliding window method as illustrated in Figure 7.9. For re-framing training data, at each iteration, we slide over 1 time step (1 day) by having overlapping windows. Whereas, for re-framing test data, we slide over one whole week to avoid data leakage (Chapter 3). Size of output window size is 1 for one-step ahead forecasting and 7 for multi-step ahead forecasting.

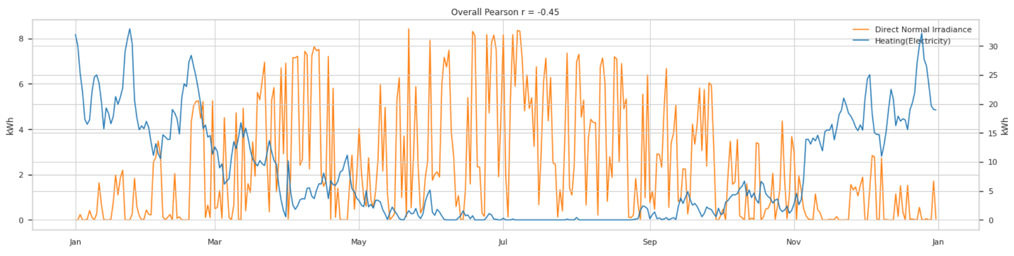
For each building, we have one-year data. We use 70% of the data for training (approximately nine months) and remaining data for testing. The data set was further scaled so all values will be between 0 and 1, using min-max normalization.

7.5.2 Experimental Metrics

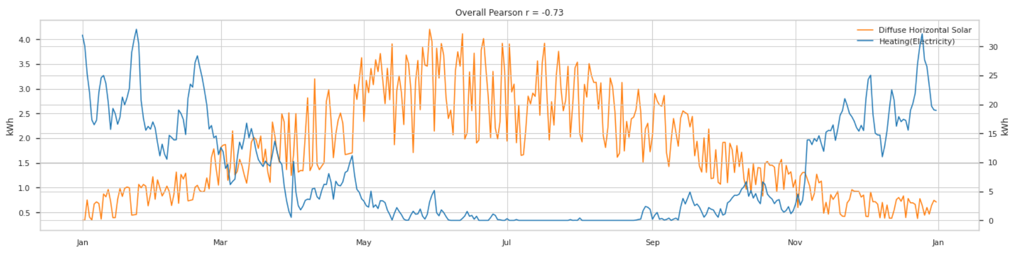
We assess our proposed predictive models using three standard performance metrics; root mean squared error (RMSE), mean area error (MAE), and coefficient of determination, also denoted as R^2. RMSE computes the square root of the mean squared difference between prediction and ground truth. MAE measures the mean absolute difference between prediction and ground truth. R^2 measures the proportion of variance explained by the prediction model to the total variance in the observed data. The value of R^2 normally ranges between 0 and 1, with 1 indicating the perfect fit. Values may, however, fall outside the range of 0 to 1 if the predictive model is worse than using an



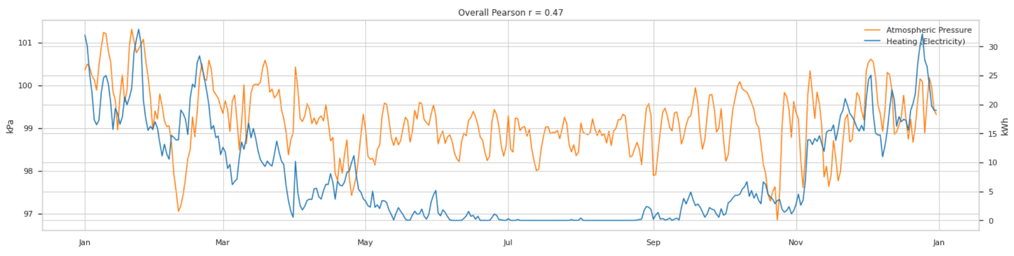
(a) Outside Air Temperature



(b) Direct Normal Irradiation



(c) Diffuse Horizontal Irradiation



(d) Atmospheric Pressure

Figure 7.8: Line plots of respectively: heating energy consumption, outside air temperature (top), and atmospheric pressure (bottom) for a given building.

horizontal hyperplane that passes through the mean value.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2}$$

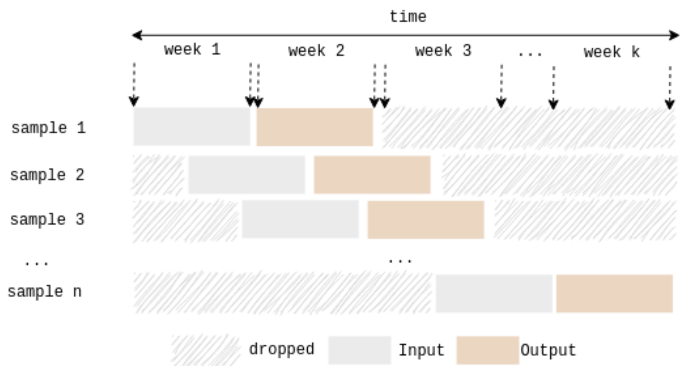


Figure 7.9: Sliding window method to re-frame time series data as supervised learning problem.

where y_i denotes the true observed value of the i-th sample, $\hat{y}_i$ denotes the predicted value of the i-th sample, $\bar{y}$ denotes the mean of the observed data, N denotes the size of the data set.

7.5.3 Evaluation with Different Scenarios

The training data recommendation based methodology was compared to two other case scenarios, which are namely *random selection* and *intra-domain* cases:

- **Recommendation** consists in our proposed methodology. Each target building from the held-out test set is provided as input (query) to our workflow. As such, a new feature embedding is obtained from the metadata vector of this target building using our trained representation learning model (see Section7.4). Most similar source buildings are subsequently identified using a simple k-nearest neighbors algorithm between target building embedding and source buildings embeddings. Trained data are finally retrieved from recommended source buildings and used to train the predictive model of the target building.
- **Random selection** consists in the case where training and test data were retrieved from respectively randomly selected source and target buildings. The case of random selection would enable us to establish that, predictive modeling built on training data from recommended similar buildings is significantly more accurate than randomly selecting data without regard to inter-building similarity.
- **Intra-domain** consists on training and testing on the same building data, hence the same domain. Intra-domain therefore stands for the classical machine learning framework in the context of building energy predictive modeling. The case of intra-domain learning would establish the relevance of the experimental results of our proposed methodology when compared to the classical machine learning framework, in which training and test data are sampled from the same domain.

Note that each case is executed 21 times (without replacement), which corresponds to the total number of held-out (unseen) target buildings used for testing. As such, each case is executed for each query, and hence each unseen target. building This ensures reliable evaluation of our methodology.

Given that we dispose of one year energy data for each building in our experimental dataset, we train predictive models using two source buildings (in random selection and recommendation-based scenarios). This way, we will form a training dataset that spans approximately 2 years. For intra-domain scenario, no source buildings are used, given that the training and test are performed on the same target building.

7.5.4 Setting 1: One-step Ahead Prediction

For one-step ahead predictive modeling, evaluations are performed using four predictive models, which are a kernel SVR, a MLP, a LSTM-RNN, and a CNN. A detailed description of machine learning algorithms proposed for multivariate time series forecasting can be found in Chapter 3.

In brief, SVR formulates and solves the regression problem using convex optimization. SVR can be extended to solve non-linear regression problems using kernel tricks. MLP represents the most basic form of artificial neural networks. A MLP is created by "stacking" several fully-connected linear layers of neurons, with non-linear activation functions in between them. The output of one layer is the input of the subsequent layer. The output of the last layer gives the final prediction of the learning model. MLP design does not intuitively take into account multiple temporal variables as in RNN. Therefore, when processing multivariate time series, we can simply flatten multi-dimensional input time series into one input vector. Otherwise, we can use a multi-head MLP (MH-MLP). This way, each univariate time series is handled by an individual head and outputs of each head are subsequently combined to give the final predictions.

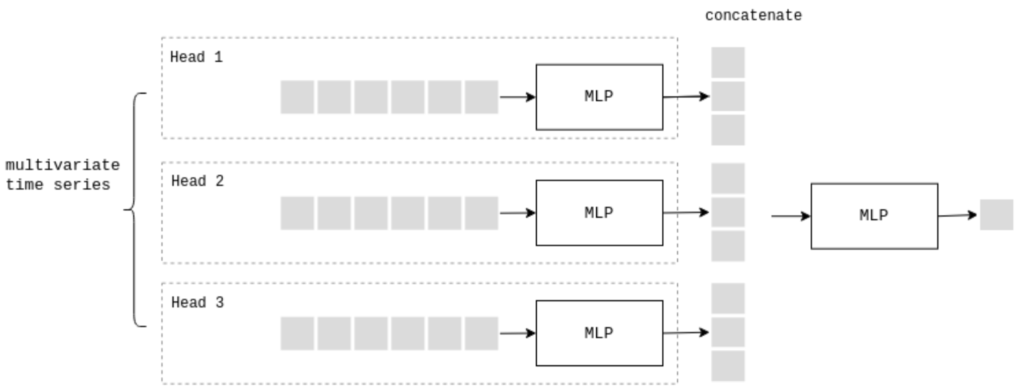


Figure 7.10: Illustration of the architecture of multi-head MLP.

RNN has been historically proposed for sequence modeling tasks. It was therefore naturally applied to time series forecasting. The reason behind the effectiveness of RNN comes from its ability to capture past information from data, and use it to inform upcoming sequence steps. LSTM is an RNN architecture. It offers the ability to capture long-term dependencies, while addressing vanishing/exploding gradients problem [192] often encountered in recurrent networks. This is essentially achieved via special gates that determine which past information should be passed and which should be forgotten [89].

Time series can be defined as one-dimensional vectors. Hence, one-dimensional convolutions are applied and slided over time dimension to extract relevant features. For processing multivariate time series, we can use either multi-channel CNN (MC-CNN) or multi-head CNN (MH-CNN). Both multi-channel and multi-head CNN process multivariate time series as multiple univariate time series. In multi-channel CNN each univariate time series is processed a separate channel, whereas, in multi-head CNN, each univariate time series is processed by a separate sub-model.

Hereafter, we detail their respective hyper-parameters. We empirically explored variants of each models architecture to fine-tune their hyperparameters, and eventually retained the settings that yielded the best evaluation results.

1. **kernel SVR** Regularization parameter C is fixed to 1. Margin of tolerance $\epsilon = 0.1$. We use a polynomial kernel of degree 5.

2. **MLP** In case of MLP model, lag observations must be flattened into features. Therefore, the input layer will correspond to a 28-dimensional feature vector (sequence length$\times$number of input features), whereas the output layer is composed of a single neuron. MLP network is composed of one hidden layer of size 100.

3. **LSTM-RNN** The input sequence is of length 14. The input layer accepts a 6-dimensional feature vector, whereas the output layer is composed of a single neuron. Our network is composed of two hidden layers; one LSTM layer of size 8, and one fully-connected layer of size 16.

4. **Multi-channel CNN** This model is composed of 5 layers in total. We detail the architecture of the trained multi-channel CNN model in Table 7.2. First input layer is of shape $[\text{BATCH} \times 14 \times 6]$, where BATCH corresponds to the batch size which is set depending on the number of samples fed to the model during training or inference, 14 corresponds to the number of time series steps, and 6 corresponds to the number of time series variables (i.e. number of channels). The subsequent one-dimensional convolution layer applies 64 filters of length 3 ($[1 \times 3]$), with no padding and stride of length 1. This results in 64 feature maps of size $[\text{BATCH} \times 12]$. The one-dimensional max-pooling in the convolution layer has a stride of 2 ($[1 \times 2]$). Therefore, as output, the dimension of feature maps is divided by 2, i.e. $12/2 = 6$. The subsequent fully-connected layer is of size 50. Given that we are performing a one-step ahead time series forecasting, the output fully-connected layer is of size 1.

5. **Multi-head CNN** We process each univariate time series using 6 convolution head. We detail the architecture of the trained multi-head CNN model in Figure 7.11. Each convolution head is composed of an input layer, two convolution

layers, and a flattening layer. Input layer is of shape $[14 \times 1]$. First convolution layer applies 32 filters of length 3. Max-pooling is applied in second convolution layer with a stride of 2. We subsequently concatenate the output of the convolution heads and apply three fully-connected layers. First and second fully connected layer is of respectively size 200 and 100, whereas output fully-connected layer is of size 1.

Table 7.2: Multi-channel CNN model architecture. A.F stands for activation function. Max-pool. stands for max-pooling.

#	Layer type	Units	A.F	Kernel shape	Max-pool.	Output
1	Input	-	-	-	-	(- x 14)x6
2	Convolutional	64	ReLU	3	2	$(- \times 6) \times 64$
3	Flattening	-	-	-	-	$(- \times 384)$
4	MLP	50	-	-	-	$(- \times 50)$
5	MLP	1	-	-	-	$(- \times 1)$

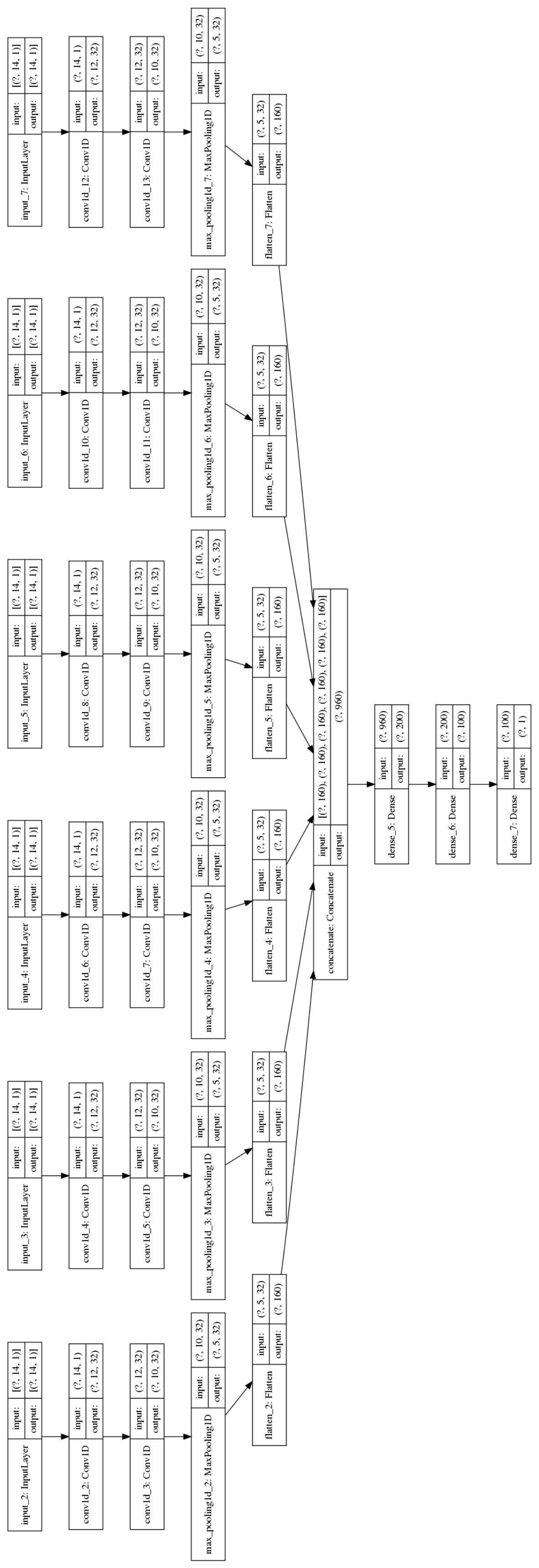

input_2: InputLayer | input: [(?, 14, 1)] | output: [(?, 14, 1)]
conv1d_2: Conv1D | input: (?, 14, 1) | output: (?, 12, 32)
conv1d_3: Conv1D | input: (?, 12, 32) | output: (?, 10, 32)
max_pooling1d_2: MaxPooling1D | input: (?, 10, 32) | output: (?, 5, 32)
flatten_2: Flatten | input: (?, 5, 32) | output: (?, 160)
input_3: InputLayer | input: [(?, 14, 1)] | output: [(?, 14, 1)]
conv1d_4: Conv1D | input: (?, 14, 1) | output: (?, 12, 32)
conv1d_5: Conv1D | input: (?, 12, 32) | output: (?, 10, 32)
max_pooling1d_3: MaxPooling1D | input: (?, 10, 32) | output: (?, 5, 32)
flatten_3: Flatten | input: (?, 5, 32) | output: (?, 160)
input_4: InputLayer | input: [(?, 14, 1)] | output: [(?, 14, 1)]
conv1d_6: Conv1D | input: (?, 14, 1) | output: (?, 12, 32)
conv1d_7: Conv1D | input: (?, 12, 32) | output: (?, 10, 32)
max_pooling1d_4: MaxPooling1D | input: (?, 10, 32) | output: (?, 5, 32)
flatten_4: Flatten | input: (?, 5, 32) | output: (?, 160)
input_5: InputLayer | input: [(?, 14, 1)] | output: [(?, 14, 1)]
conv1d_8: Conv1D | input: (?, 14, 1) | output: (?, 12, 32)
conv1d_9: Conv1D | input: (?, 12, 32) | output: (?, 10, 32)
max_pooling1d_5: MaxPooling1D | input: (?, 10, 32) | output: (?, 5, 32)
flatten_5: Flatten | input: (?, 5, 32) | output: (?, 160)
input_6: InputLayer | input: [(?, 14, 1)] | output: [(?, 14, 1)]
conv1d_10: Conv1D | input: (?, 14, 1) | output: (?, 12, 32)
conv1d_11: Conv1D | input: (?, 12, 32) | output: (?, 10, 32)
max_pooling1d_6: MaxPooling1D | input: (?, 10, 32) | output: (?, 5, 32)
flatten_6: Flatten | input: (?, 5, 32) | output: (?, 160)
input_7: InputLayer | input: [(?, 14, 1)] | output: [(?, 14, 1)]
conv1d_12: Conv1D | input: (?, 14, 1) | output: (?, 12, 32)
conv1d_13: Conv1D | input: (?, 12, 32) | output: (?, 10, 32)
max_pooling1d_7: MaxPooling1D | input: (?, 10, 32) | output: (?, 5, 32)
flatten_7: Flatten | input: (?, 5, 32) | output: (?, 160)
concatenate: Concatenate | input: [(?, 160), (?, 160), (?, 160), (?, 160), (?, 160), (?, 160)] | output: (?, 960)
dense_5: Dense | input: (?, 960) | output: (?, 200)
dense_6: Dense | input: (?, 200) | output: (?, 100)
dense_7: Dense | input: (?, 100) | output: (?, 1)

Figure 7.11: Multi-head CNN model architecture.

For neural networks, namely MLP, LSTM-RNN and CNN models, the output layer consists of a fully-connected layer with linear activation function. The Rectified Linear Unit (ReLU) is used as the non-linear activation function for hidden layers. Fine-tuning of weights is done using Adam optimization algorithm [128]. Adam (adaptive moment estimation) optimization is an extension of classic stochastic gradient descent (SGD). Adam is proven to be computationally efficient and fast coverging [128]. Adam computes individual adaptive learning rates for different parameters instead of manually setting it as in SGD.

Weights initialization follows a normal distribution with zero mean and standard deviation $\sigma = 1$, whereas biases are initialized to zeroes. The gradients are back-propagated through batches of length 80. For the training epochs number, we have fixed 1000 as the maximum number of epochs. To avoid over-fitting, we have implemented an early stopping mechanism which breaks the training loop when training cost does not improve on the training set after 20 epochs.

7.5.5 Setting 2: Multi-step Ahead Prediction

For multi-step ahead predictive modeling, we seek to forecast daily energy consumption for the subsequent week given data about previous two weeks. We use the same input data as in one-step ahead predictive modeling task. As machine learning algorithms, we leverage multi-output SVR, multi-head MLP, LSTM-RNN and CNN. Details about machine learning models proposed in the context of multi-step ahead time series forecasting are provided in Chapter 3.

Multi-output SVR is a generalization of the classic single-output SVR. It consists in iteratively training multiple SVR models, where each individual model predicts a time step of the prediction horizon. As CNN, we test both multi-head and multi-channel CNNs. However, by contrast to one-step ahead time series forecasting, we simply modify output layer to serve a vector instead of one value.

We may use the seq2seq architecture for LSTM-RNN to predict multiple future time steps. Seq2seq consists in an encoder-decoder architecture, which uses two sub-networks: an encoder reads the input sequence and summarizes useful information into a context vector, and a decoder that read this context vector and predicts the target sequence. An attention mechanism is also used on the encoder outputs at each decoding step as in [13]. In addition, we test hybrid CNN and LSTM-RNN model that couple both of these approaches. We refer to this model as ConvLSTM. This model benefits from both CNN feature extraction and LSTM sequential data processing capabilities.

We detail hyperparameters used for different algorithms hereafter. As in previous experiments, we retain those that yielded best prediction results in each algorithm.

1. **Multi-output kernel SVR** We use the same architecture as in the previous section for each constituent single-output SVR model. Regularization parameter $C = 1$. Margin of tolerance $\epsilon = 0.1$. We use a polynomial kernel of degree 5.

2. **Multi-head MLP** The network is composed of 7 MLP heads, each corresponding to an input time series variable. Heads are composed of a two fully-connected layer of respectively size 200 and 100. Outputs of separate heads are merged and fed to a MLP composed of two hidden layers of respectively size 200 and 100.

3. **Seq2seq** For encoding the input sequence, we use a single LSTM layer of size 200. For decoding, we use an LSTM layer 200, followed by a fully-connected layer of size 100, and the output layer. An attention mechanism is adopted on the encoder outputs.

4. **Multi-channel CNN** We use 5 hidden layers in total, namely three 1D convolution layers, a flattening layer and one fully-connected (MLP) layer. First and second convolution layer use 32 filters of size 3 and zero padding, whereas, third layer uses 16 filters, and contains max-pooling with a stride of 2. Flattening layer transforms resulting 3-dimensional vector into a 2-dimensional vector. This vector is served to subsequent fully-connected layer, which contains 100 units.

5. **Multi-head CNN** As in one-step ahead setting, the trained network is composed of 7 heads, each corresponding to input time series variables. Each head is composed of two 1D convolution layers. Both use 32 filters of size 3 and no padding. Second convolution layer includes in addition a max-pooling operation with a stride of 2. Outputs of the CNN heads are subsequently concatenated, and passed through 2 fully connected hidden layers containing respectively 200 and 100 units.

6. **ConvLSTM** For the encoder, we use three 1D convolution layers. First two layers use 32 filters of length 3 and zero padding. Third layer use 16 filters of size 3 and zero padding. This layer include a max-pooling operation with a stride of 2. Finally, a flattening layer transforms the results to a 2D vector. For the decoder, we use one LSTM layer followed by a hidden and an output fully-connected layers. The LSTM layer contains 200 units, whereas, the following hidden fully-connected layer contains 100 units.

As in one-step ahead predictive modeling, for neural networks, we use linear activation function in output layers and ReLU activation function in hidden layers. Optimization of weights is conducted using Adam algorithm.

7.5.6 Experimental Results

The experimental results are shown for respectively one-step ahead time series forecasting in table 7.3, and multi-step ahead time series forecasting in table 7.4. In both experiments, we provide average errors and standard deviations across all 21 executions (number of held out test buildings), for each case scenario and each model. In Figure 7.12, we further detail results of one-step ahead predictive modeling using a box and whisker plot which shows the distribution of experimental results and skewness.

Table 7.3: Evaluation results of one-step ahead predictive modeling. Bold font indicates best results obtained across each evaluation metrics for each model and for each test case. "_" stands for R^2 values that fall outside of the range of 0% to 100%.

Model	Test Case	MAE (kWh)		RMSE (kWh)		R^2 (%)	
		Mean	*Std*	*Mean*	*Std*	*Mean*	*Std*
MLP	Random selection	12.89	6.95	16.87	8.70	_	_
	intra-Domain	5.05	4.24	6.64	5.41	88.64	10.58
	Recommendation	**3.95**	**3.25**	**5.18**	**4.22**	**91.74**	**5.67**
RNN-LSTM	Random selection	13.91	12.62	17.89	15.04	_	_
	intra-Domain	6.84	5.14	9.60	7.22	74.11	22.87
	Recommendation	**3.93**	**3.67**	**5.23**	**4.72**	**91.26**	**6.03**
MH-CNN	Random selection	25.08	14.58	33.14	19.59	_	_
	intra-Domain	8.78	8.07	11.36	10.15	67.33	41.24
	Recommendation	**2.77**	**2.89**	**3.40**	**3.44**	**93.80**	**6.28**
MC-CNN	Random selection	22.71	15.71	30.03	20.63	_	_
	intra-Domain	4.91	5.93	5.87	6.71	80.97	23.30
	Recommendation	**2.78**	**2.65**	**3.47**	**3.21**	**92.85**	**6.77**
SVR-Kernel	Random selection	17.62	20.84	34.26	45.86	_	_
	intra-Domain	8.69	7.57	13.37	10.99	62.80	16.72
	Recommendation	**3.21**	**3.20**	**5.70**	**5.39**	**92.37**	**6.08**
Proposed Ensemble	Recommendation	**2.06**	**2.21**	**3.27**	**3.46**	**97.82**	**1.39**

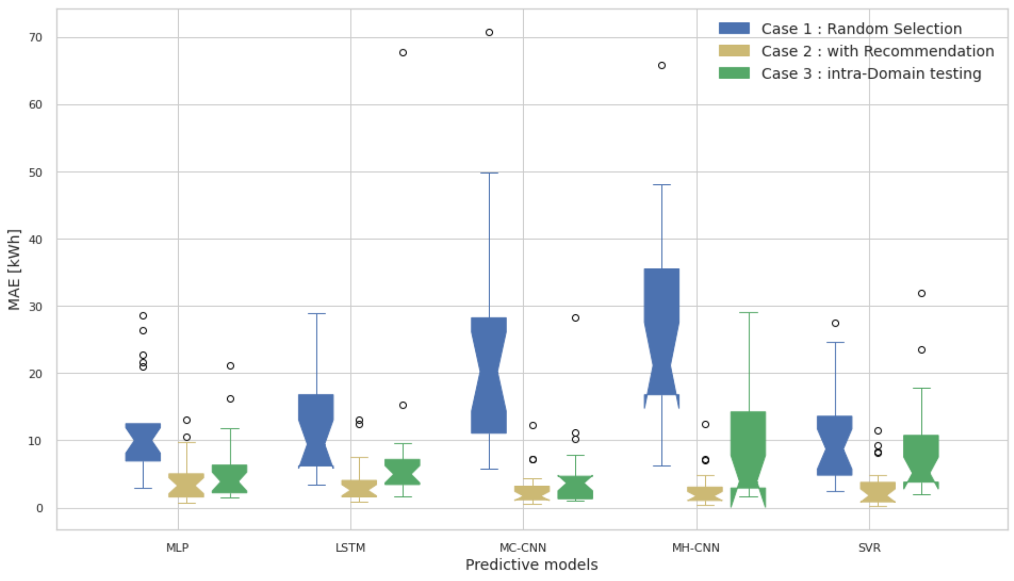


Figure 7.12: Boxplot of the experimental results grouped for each predictive model and for each evaluation scenario. The whiskers illustrate the ranges for the lower 25% and the upper 25% of the experimental results, excluding outliers. Outliers are identified by circles (○).

Table 7.4: Evaluation results of multi-step ahead predictive modeling. Bold font indicates best results obtained across each evaluation metrics for each model and for each test case. "_" stands for R^2 values that fall outside of the range of 0% to 100%.

Model	Test Case	MAE (kWh)		RMSE (kWh)		R^2 (%)	
		Mean	*Std*	*Mean*	*Std*	*Mean*	*Std*
MH-MLP	Random selection	14.82	10.44	20.44	14.21	_	_
	intra-Domain	10.97	9.84	15.49	13.80	48.11	57.54
	Recommendation	**2.97**	**2.90**	**4.96**	**5.09**	**92.61**	**7.53**
Seq2seq	Random selection	15.20	10.83	21.27	12.59	_	_
	intra-Domain	13.03	9.29	18.06	15.25	48.28	34.80
	Recommendation	**2.95**	**2.88**	**4.96**	**5.16**	**92.0**	**8.46**
MH-CNN	Random selection	15.10	8.55	21.81	11.86	_	_
	intra-Domain	10.58	7.42	14.10	9.08	47.15	27.17
	Recommendation	**3.33**	**3.22**	**5.54**	**5.31**	**91.16**	**8.40**
MC-CNN	Random selection	26.14	15.86	38.02	22.94	_	_
	intra-Domain	8.54	7.78	11.93	10.02	62.07	45.76
	Recommendation	**3.46**	**2.94**	**5.73**	**5.04**	**90.26**	**7.50**
ConvLSTM	Random selection	24.89	17.48	35.93	25.53	_	_
	intra-Domain	11.27	8.81	15.11	11.55	45.11	40.47
	Recommendation	**3.22**	**2.97**	**5.40**	**5.13**	**91.19**	**7.82**
MO-SVR	Random selection	17.06	12.84	22.25	17.94	_	_
	intra-Domain	11.91	11.08	15.95	15.32	53.60	18.10
	Recommendation	**3.89**	**4.38**	**6.57**	**6.62**	**90.13**	**8.32**
Proposed Ensemble	Recommendation	**2.78**	**2.85**	**4.60**	**4.97**	**93.02**	**7.40**

One-step ahead Given Table 7.3 and Figure 7.12, best generalization results are provided by kernel SVR model. However, as noticed, averaged evaluation results are approximately similar across studied prediction models. More detailed visualization of experimental results for all 21 executions is depicted in Figure 7.13. This plot shows that there is no specific predictive modeling algorithm that outperforms all other algorithms in every scenario. As such, MC-CNN and MH-CNN models hold best for test buildings number 2, 9 and 18. However, they hold worse performance for test buildings 3 and 17. Similarly, SVR model performs best for test buildings 11 and 21, while worst for test building 9. This phenomenon may be explainable by the no free lunch (NFL) theorem (see Chapter 6). As a result, we also resorted to combining several algorithms using stacked generalization technique. Ensemble model allows to have the best performance as shown in table 7.3.

Multi-step ahead From Table 7.4, we notice that best generalization results are obtained with recommendation-based scenario. Among predictive modeling algorithms,

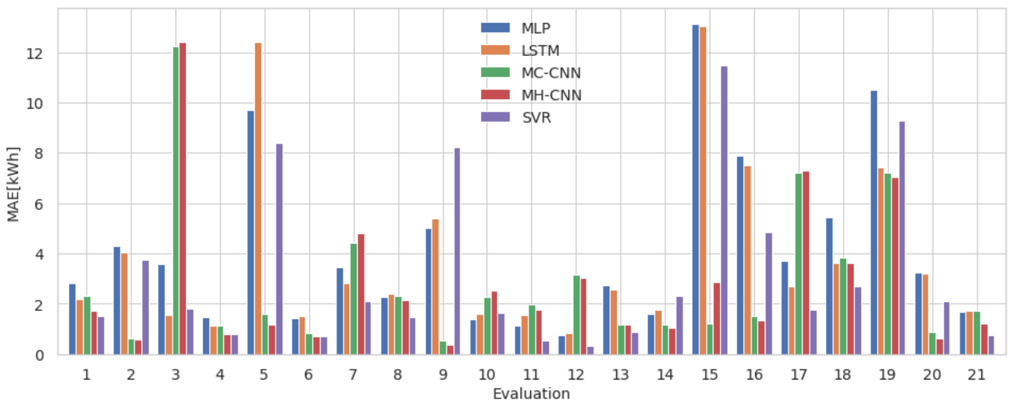


Figure 7.13: Barplot of the experimental results of our recommended methodology for each predictive model and for each evaluation iteration.

we notice that Seq2seq with attention mechanism and MH-MLP perform the best compared to other models. Similarly to one-step ahead setting, we also notice great variation in performance across all metrics in intra-domain scenario. Ensemble learning model further improves generalization performance and yield an R^2 of 93%.

7.5.7 Discussion

Overall, comparison shows that the recommendation-based methodology exhibits better generalization performance compared to both intra-domain learning and random selection cases, for all models in one-step ahead and multi-step ahead predictive modeling.

We can therefore conclude that the predictive model factory has enabled us to train robust models that perform better, on average, in cross-domain setting than in intra-domain setting. The hypothesis is that this is due to the fact that we are training disparate models with fixed hyper-parameters across all executions. Hyper-parameters were therefore not fine-tuned and optimized for each training set, as traditionally done in a classical machine learning frameworks. Generally, a model with fixed hyper-parameters is less likely to have optimal performance across all training sets (across all executions), which are retrieved from different buildings. However, our methodology delivered stable near-optimal results in almost every trial.

Therefore, this workflow offers a generic methodology for cross-building predictive modeling by recommendation of training data retrieved from similar buildings. Predictive modeling workflow is proven to output models that are generalizable and performs relatively well independently of target building.

7.6 Summary and conclusions

In this chapter, we evaluate our predictive model factory for the use case of heating energy prediction in buildings. Experimental dataset was generated using simulation tool DesignBuilder. Heterogeneity within simulated data is established through random setting of several parameters in building models during augmentation process. This ensures a reduced risk of bias in our experimental evaluations. As building metadata, several contextual features are considered, namely activity sector, typology, location, occupation and construction properties. Evaluations were conducted on similar buildings recommendation process and predictive modeling process of our methodology.

For the process of buildings recommendation, a feature representation model is used to extract new task-specific features from buildings metadata. For this, a deep similarity metric learning is performed using a Siamese multi-layer perceptrons. Evaluation of recommendations shows that source buildings selection based on deep similarity learning offers more relevant recommendations to target buildings, when compared to simple metadata-based selection.

For the predictive modeling process, two experimental settings are considered: one-step ahead and multi-step ahead predictions. In the one-step ahead setting, used ML algorithms are kernel SVR, MLP, LSTM-RNN, multi-channel CNN and multi-head CNN. Best average prediction results of our recommendation-based methodology are obtained using CNN models. In the multi-step ahead setting, we also use a multi-output kernel SVR, a multi-head MLP, a sequence-to-sequence LSTM-RNN, a multi-head CNN, a multi-channel CNN and a combination of CNN and LSTM-RNN. Best average results of our methodology are obtained using sequence-to-sequence and multi-head MLP models. Overall, for both experimental settings, and for all used ML algorithms, we demonstrate that our recommendation-based methodology offers a generic framework that performs better in average than intra-domain and random selection scenarios. Intra-domain scenario represents a classical ML setting where training and test data are sampled from the same building. Random selection scenario stands for randomly selecting training buildings from available source buildings to produce a predictive model for the held-out target building.

Finally, trained models predictions are combined using a generalization stacking framework. Experimental evaluations show that stacked generalization further improves prediction results in both one(step ahead and multi-step ahead settings.

Chapter 8

Conclusions and Directions for Future Work

The scope of this thesis is to present a methodology for predictive modeling of target buildings on which no historical data are available. To that end, predictive models are dynamically constructed using solely contextual descriptions (metadata) about the given target building. This is particularly useful in the context of newly built and newly renovated buildings.

The core idea is to leverage available historical data about other source buildings. Our methodology is two-fold. The first process recommends most similar source buildings to a target building using their contextual descriptions (metadata). Contextual description about the target building is provided as input query. Similarity distance between metadata is learned specifically to our target task of predictive modeling. Hence, buildings metadata are mapped to new discriminative feature space, in which similar buildings are mapped close to each other, and dissimilar building pairs are mapped far from each other.

A deep similarity learning framework with Siamese network is adopted. For a task-specific similarity learning, similarity measures between pairs of buildings should be based on their historical data, rather than their metadata. This is due to the fact that target predictive models are trained on historical data. Thereby, we propose to use dynamic time warping (DTW) for the training of the Siamese network. However, during inference, our recommendation workflow only requires target building metadata.

The second process trains predictive models for the target building using training data recommended by the previous process. Historical data of buildings energy are multivariate time series data. Machine learning models that are used in this context are support vector regression and artificial neural networks. As neural networks, multi-layer Perceptron (MLP), along with recurrent neural networks (RNN), convolutional neural networks (CNN), and combination of both are used. These models are subsequently combined together via stacking to ensure a robust performance.

A microservice-based implementation of our predictive model factory is designed. Logically independent workflows are designed as microservices. Thereby, four microservices are developed. The first process of our methodology is performed by two disparate microservices: similarity learning and training data recommendation. The similarity learning microservice trains feature transformation models from labeled pairwise similarity information between source buildings data. This microservice can be run periodically in background. The training data recommendation microservice is launched at each user query. It processes metadata about the target building contained in the query, and use them to recommend most similar source buildings available in our system. The second process is also performed by a single predictive modeling microservice.

In addition, we design a building data handling microservice. This microservice is responsible for integrating building-related data collected from multiple data sources. For data integration, a unified global view is used. Information about source buildings consist of historical data (time series) and metadata (contextual description). As such, historical data, consisting of energy data and weather data, are stored in corresponding time series stores. These time series data are linked together through building metadata, which are modeled based on ontologies. We extend ThinkHome ontology to describe building design and thermal parameters, and basic information about occupants.

Experimental evaluations of our predictive modeling factory is conducted using a simulated heterogeneous dataset. Buildings simulation models are generated via augmentation of a set of base buildings. Overall, the experimental results have shown that our methodology can be used effectively for the use case of cross-building heating energy forecasting. More specifically, evaluation of the first process of our methodology shows that deep similarity learning based on DTW between buildings data offers better results than directly using raw building metadata. Also, comparison between recommended source buildings using our methodology and ground truth validates its usefulness.

Evaluation of the second process of our methodology shows that it performs better than random selection and intra-domain scenarios. According to our hypothesis, this is due to the fact that we are using predictive models with fixed hyper-parameters across all test executions. Hyper-parameters were therefore not fine-tuned and optimized for each training set, as traditionally done in a classical machine learning framework. Hence, we conclude that our methodology offers a generic framework that is faster and performs better on average than classical machine learning framework (intra-domain scenario).

Different models were evaluated. Namely, we apply a multi-head MLP, multi-head and multi-channel CNNs, a seq2seq model with attention mechanism, and an hybrid CNN and LSTM-RNN model. On average, multi-head CNN offers best results for one-step ahead time series forecasting, whereas seq2seq model offers best results for multi-step time series forecasting. However, a more detailed look at our results shows that there is no single model that performs better than any other model for each test building. Hence, an ensemble learning framework is adopted via stacked generalization. This allows a more robust predictive modeling, and further contributes to the genericity and extensibility of the developed predictive model factory.

Our methodology helps to model a wide variety of target buildings, given its generic nature that allows it to dynamically adapt to target buildings using queries. More generally, our methodology should not only be considered as an answer to the use case of cross-building energy modeling, but as a possible solution to a broader range of applications. In its design, our methodology performs cross-domain knowledge transfer for a specific task, by utilizing two types of information; contextual description about entities (easily acquired metadata), and training data (with respect to the task), in a way that it requires minimal description and no training data about a new entity to model it.

8.1 Limitations and Future work

This thesis is a first step towards a larger research agenda that aims at developing a fully automated predictive model factory. We present some limitations of our work and potential directions for future work hereafter.

The short-term perspectives consist in improving different processes of our proposed methodology. Particularly, in the training data recommendation process, we perform representation learning using homogeneous buildings metadata, in terms of metadata features. Namely, we estimate that available source buildings and target buildings (described in user queries) have the same features, such as building typology, area, number of occupants, etc. It is therefore interesting to extend our methodology, and more specifically our similarity learning and training data recommendation processes, to handle multi-modal contextual descriptions on available source buildings [139]. This will add further flexibility of our system by allowing the exploitation of source buildings on which different metadata are available.

Another aspect that merits further consideration is the evolution of the deployed target predictive model. As such, in our work, the target model is trained on recommended data collected from most similar source domains. Experimental evaluation demonstrated that the proposed target models offer good results when tested on the new previously unseen target domains. This is particularly useful in cases where target data are not or are not yet available. However, it is useful to consider the case where target data are progressively made available after the predictive model deployment. This can be tackled by refining target predictive models once target data are accessible. For instance, Mancini et al. [162] have proposed a continuous refinement strategy for deep models by leveraging the incoming stream of target data.

Furthermore, we only performed experimental evaluations for the use case of building energy modeling. However, given its generic design, our methodology can be re-used for other applications in other sectors, such as healthcare and finances. As future work, it is therefore important to validate the genericity of our methodology for other use cases.

The long-term perspectives consist in investigating new research directions raised by our work. Firstly, for the purpose of our work, we suppose that collected data and metadata on source buildings are already integrated into an unified ontology-based global view. data collection [208] and integration [143] processes therefore falls beyond the scope of this research. However, it is primordial to automate these tasks to exploit existing data on buildings from multiple heterogeneous sources. As such, automated data collection involves the discovery of relevant datasets online (i.e open data), or acquiring data through simulation and/or augmentation. Automated data integration involves the definition of mappings between the sources and the unified global view.

Big data nowadays play an important role in various sectors. In the context of predictive modeling with time series data, big data offer great opportunities in decision making, as well as numerous computational challenges [137]. It is therefore interesting to investigate computational efficiency and scalability of our system, and its constituent microservices, when processing larger-scale data.

Bibliography

[1] H. Abdi and L. J. Williams. “Principal component analysis”. In: *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), pp. 433–459.

[2] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin. *Learning from data*. Vol. 4. AMLBook New York, NY, USA: 2012.

[3] C. Achille, C. Tommasi, et al. “Interoperability matter: Levels of data sharing, starting from a 3D information modelling”. In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 42 (2017), pp. 623–630.

[4] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. “A learning algorithm for Boltzmann machines”. In: *Cognitive science* 9.1 (1985), pp. 147–169.

[5] M. Aksoezen, M. Daniel, U. Hassler, and N. Kohler. “Building age as an indicator for energy consumption”. In: *Energy and Buildings* 87 (2015), pp. 74–86.

[6] A. N. Alfiyatin, R. E. Febrita, H. Taufiq, and W. F. Mahmudy. “Modeling house price prediction using regression analysis and particle swarm optimization”. In: *International Journal of Advanced Computer Science and Applications* 8.10 (2017), pp. 323–326.

[7] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari. “The history began from alexnet: A comprehensive survey on deep learning approaches”. In: *arXiv preprint arXiv:1803.01164* (2018).

[8] K. Amasyali and N. M. El-Gohary. “A review of data-driven building energy consumption prediction studies”. In: *Renewable and Sustainable Energy Reviews* 81 (2018), pp. 1192–1205.

[9] R. Apanaviciene, A. Vanagas, and P. A. Fokaides. “Smart Building Integration into a Smart City (SBISC): Development of a New Evaluation Framework”. In: *Energies* 13.9 (2020), p. 2190.

[10] Y. Arayici, J. Counsell, L. Mahdjoubi, G. A. Nagy, S. Hawas, and K. Dweidar. *Heritage building information modelling*. Taylor & Francis, 2017.

[11] J. S. Armstrong and F. Collopy. “Error measures for generalizing about forecasting methods: Empirical comparisons”. In: *International journal of forecasting* 8.1 (1992), pp. 69–80.

[12] J. Aßfalg, H.-P. Kriegel, P. Kröger, P. Kunath, A. Pryakhin, and M. Renz. “Similarity search on time series based on threshold queries”. In: *International Conference on Extending Database Technology*. Springer. 2006, pp. 276–294.

[13] D. Bahdanau, K. Cho, and Y. Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).

[14] B. Balaji, A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, J. Ploennigs, Y. Agarwal, et al. “Brick: Metadata schema for portable smart building applications”. In: *Applied energy* 226 (2018), pp. 1273–1292.

[15] B. Balaji, A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, J. Ploennigs, Y. Agarwal, et al. “Brick: Towards a unified metadata schema for buildings”. In: *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*. 2016, pp. 41–50.

[16] Y. Balaji, S. Sankaranarayanan, and R. Chellappa. “Metareg: Towards domain generalization using meta-regularization”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 998–1008.

[17] K. Bandara, C. Bergmeir, and S. Smyl. “Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach”. In: *Expert systems with applications* 140 (2020), p. 112896.

[18] Y. Bao, T. Xiong, and Z. Hu. “Multi-step-ahead time series prediction using multiple-output support vector regression”. In: *Neurocomputing* 129 (2014), pp. 482–493.

[19] J. R. Bedell and N. Kohler. “A hierarchical model for building applications”. In: (1993).

[20] A. Bellet, A. Habrard, and M. Sebban. “A survey on metric learning for feature vectors and structured data”. In: *arXiv preprint arXiv:1306.6709* (2013).

[21] R. Bellman. “Dynamic programming”. In: *Science* 153.3731 (1966), pp. 34–37.

[22] A. E. Ben-Nakhi and M. A. Mahmoud. “Cooling load prediction for buildings using general regression neural networks”. In: *Energy Conversion and Management* 45.13-14 (2004), pp. 2127–2141.

[23] Y. Bengio, A. Courville, and P. Vincent. “Representation learning: A review and new perspectives”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

[24] H. S. Bhatt, A. Rajkumar, and S. Roy. “Multi-Source Iterative Adaptation for Cross-Domain Classification”. In: *IJCAI*. 2016, pp. 3691–3697.

[25] S. Bhattacharjee and G. Reichard. “Socio-economic factors affecting individual household energy consumption: A systematic review”. In: *Energy Sustainability*. Vol. 54686. 2011, pp. 891–901.

[26] S. Bickel and T. Scheffer. “Dirichlet-enhanced spam filtering based on biased samples”. In: *Advances in neural information processing systems*. 2007, pp. 161–168.

[27] M. R. Biswas, M. D. Robinson, and N. Fumo. “Prediction of residential building energy consumption: A neural network approach”. In: *Energy* 117 (2016), pp. 84–92.

[28] G. Blanchard, G. Lee, and C. Scott. “Generalizing from several related classification tasks to a new unlabeled sample”. In: *Advances in neural information processing systems*. 2011, pp. 2178–2186.

[29] J. Blitzer, D. P. Foster, and S. M. Kakade. "Zero-shot domain adaptation: A multi-view approach". In: *Tech. Rep. TTI-TR-2009-1* (2009).

[30] P. M. Bluyssen. *The indoor environment handbook: how to make buildings healthy and comfortable*. Routledge, 2009.

[31] D. Bonino and F. Corno. "Dogont-ontology modeling for intelligent domotic environments". In: *International Semantic Web Conference*. Springer. 2008, pp. 790–803.

[32] G. Bontempi. "Long term time series prediction with multi-input multi-output local learning". In: *Proc. 2nd ESTSP* (2008), pp. 145–154.

[33] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola. "Integrating structured biological data by kernel maximum mean discrepancy". In: *Bioinformatics* 22.14 (2006), e49–e57.

[34] B. E. Boser, I. M. Guyon, and V. N. Vapnik. "A training algorithm for optimal margin classifiers". In: *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, pp. 144–152.

[35] A. Botchkarev. "Performance Metrics (Error Measures) in Machine Learning Regression, Forecasting and Prognostics: Properties and Typology". In: *arXiv preprint arXiv:1809.03006* (2018).

[36] M. Bourdeau, X. qiang Zhai, E. Nefzaoui, X. Guo, and P. Chatellier. "Modeling and forecasting building energy consumption: A review of data-driven techniques". In: *Sustainable Cities and Society* 48 (2019), p. 101533.

[37] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. "Domain separation networks". In: *Advances in Neural Information Processing Systems*. 2016, pp. 343–351.

[38] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[39] M. Bravetti, S. Giallorenzo, J. Mauro, I. Talevi, and G. Zavattaro. "A formal approach to microservice architecture deployment". In: *Microservices*. Springer, 2020, pp. 183–208.

[40] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. "Signature verification using a" siamese" time delay neural network". In: *Advances in neural information processing systems*. 1994, pp. 737–744.

[41] V. Bui, J. Kim, Y. M. Jang, et al. "Power Demand Forecasting Using Long Short-Term Memory Neural Network based Smart Grid". In: *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. IEEE. 2020, pp. 388–391.

[42] M. Canizo, I. Triguero, A. Conde, and E. Onieva. "Multi-head CNN–RNN for multi-time series anomaly detection: An industrial case study". In: *Neurocomputing* 363 (2019), pp. 246–260.

[43] M. Chang, Y. Lou, and L. Qiu. "An approach for time series similarity search based on Lucene". In: *2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS)*. IEEE. 2016, pp. 210–214.

[44] R. Chattopadhyay, Q. Sun, W. Fan, I. Davidson, S. Panchanathan, and J. Ye. “Multisource domain adaptation and its application to early detection of fatigue”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6.4 (2012), p. 18.

[45] S. Chaudhari, V. Mithal, G. Polatkan, and R. Ramanath. “An attentive survey of attention models”. In: *arXiv preprint arXiv:1904.02874* (2019).

[46] B. Chen, Y. Wang, R. Wang, Z. Zhu, L. Ma, X. Qiu, and W. Dai. “The gray-box based modeling approach integrating both mechanism-model and data-model: The case of atmospheric contaminant dispersion”. In: *Symmetry* 12.2 (2020), p. 254.

[47] L. Chen and R. Ng. “On the marriage of lp-norms and edit distance”. In: *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. 2004, pp. 792–803.

[48] L. Chen, M. T. Özsu, and V. Oria. “Robust and fast similarity search for moving object trajectories”. In: *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. 2005, pp. 491–502.

[49] Q. Chen, Y. Liu, Z. Wang, I. Wassell, and K. Chetty. “Re-weighted adversarial adaptation network for unsupervised domain adaptation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7976–7985.

[50] Y.-T. Chen. “The factors affecting electricity consumption and the consumption characteristics in the residential sector—A case example of Taiwan”. In: *Sustainability* 9.8 (2017), p. 1484.

[51] F. Chollet et al. *Keras*. `https://keras.io`. 2015.

[52] S. Chopra, R. Hadsell, and Y. LeCun. “Learning a similarity metric discriminatively, with application to face verification”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 1. IEEE. 2005, pp. 539–546.

[53] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. “The loss surfaces of multilayer networks”. In: *Artificial intelligence and statistics*. PMLR. 2015, pp. 192–204.

[54] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *arXiv preprint arXiv:1412.3555* (2014).

[55] Y. G. Cinar, H. Mirisaee, P. Goswami, E. Gaussier, A. Aıt-Bachir, and V. Strijov. “Position-based content attention for time series forecasting with sequence-to-sequence rnns”. In: *International conference on neural information processing*. Springer. 2017, pp. 533–544.

[56] C. Cortes, Y. Mansour, and M. Mohri. “Learning Bounds for Importance Weighting.” In: *Nips*. Vol. 10. Citeseer. 2010, pp. 442–450.

[57] C. Cortes and V. Vapnik. “Support-vector networks”. In: *Machine learning* 20.3 (1995), pp. 273–297.

[58] R. Crawford. *Life cycle assessment in the built environment*. Taylor & Francis, 2011.

[59] D. B. Crawley, L. K. Lawrie, F. C. Winkelmann, W. F. Buhl, Y. J. Huang, C. O. Pedersen, R. K. Strand, R. J. Liesen, D. E. Fisher, M. J. Witte, et al. "EnergyPlus: creating a new-generation building energy simulation program". In: *Energy and buildings* 33.4 (2001), pp. 319–331.

[60] G. Csurka. "Domain adaptation for visual applications: A comprehensive survey". In: *arXiv preprint arXiv:1702.05374* (2017).

[61] G. Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[62] H. Daumé III. "Frustratingly easy domain adaptation". In: *arXiv preprint arXiv:0907.1815* (2009).

[63] C. Deb, F. Zhang, J. Yang, S. E. Lee, and K. W. Shah. "A review on time series forecasting techniques for building energy consumption". In: *Renewable and Sustainable Energy Reviews* 74 (2017), pp. 902–924.

[64] O. Delalleau and Y. Bengio. "Shallow vs. deep sum-product networks". In: *Advances in neural information processing systems*. 2011, pp. 666–674.

[65] E. Delzendeh, S. Wu, A. Lee, and Y. Zhou. "The impact of occupants' behaviours on building energy analysis: A research review". In: *Renewable and Sustainable Energy Reviews* 80 (2017), pp. 1061–1071.

[66] T. G. Dietterich et al. "Ensemble learning". In: *The handbook of brain theory and neural networks* 2 (2002), pp. 110–125.

[67] 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". In: *Proceedings of the VLDB Endowment* 1.2 (2008), pp. 1542–1552.

[68] Z. Ding and Y. Fu. "Deep domain generalization with structured low-rank constraint". In: *IEEE Transactions on Image Processing* 27.1 (2017), pp. 304–313.

[69] P. Domingos. "A few useful things to know about machine learning". In: *Communications of the ACM* 55.10 (2012), pp. 78–87.

[70] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. "Decaf: A deep convolutional activation feature for generic visual recognition". In: *International conference on machine learning*. 2014, pp. 647–655.

[71] Q. Dou, D. C. de Castro, K. Kamnitsas, and B. Glocker. "Domain generalization via model-agnostic learning of semantic features". In: *Advances in Neural Information Processing Systems*. 2019, pp. 6450–6461.

[72] N. Dragoni, I. Lanese, S. T. Larsen, M. Mazzara, R. Mustafin, and L. Safina. "Microservices: How to make your application scale". In: *International Andrei Ershov Memorial Conference on Perspectives of System Informatics*. Springer. 2017, pp. 95–104.

[73] S. Du, T. Li, Y. Yang, and S.-J. Horng. "Multivariate time series forecasting via attention-based encoder–decoder framework". In: *Neurocomputing* 388 (2020), pp. 269–279.

[74] L. Duan, D. Xu, and S.-F. Chang. "Exploiting web images for event recognition in consumer videos: A multiple source domain adaptation approach". In: *2012*

IEEE Conference on Computer Vision and Pattern Recognition. IEEE. 2012, pp. 1338–1345.

[75] I. Esnaola-Gonzalez, J. Bermúdez, I. Fernandez, and A. Arnaiz. "Ontologies for observations and actuations in buildings: A survey". In: *Semantic Web* Preprint (2020), pp. 1–29.

[76] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. "Fast subsequence matching in time-series databases". In: *ACM Sigmod Record* 23.2 (1994), pp. 419–429.

[77] C. Fan, Y. Zhang, Y. Pan, X. Li, C. Zhang, R. Yuan, D. Wu, W. Wang, J. Pei, and H. Huang. "Multi-horizon time series forecasting with temporal attention learning". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 2527–2535.

[78] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* 33.4 (2019), pp. 917–963.

[79] S. J. Fenves, U. Flemming, C. Hendrickson, M. L. Maher, and G. Schmitt. "Integrated software environment for building design and construction". In: *Computer-Aided Design* 22.1 (1990), pp. 27–36.

[80] S. Ferlito, M. Atrigna, G. Graditi, S. De Vito, M. Salvato, A. Buonanno, and G. Di Francia. "Predictive models for building's energy consumption: An Artificial Neural Network (ANN) approach". In: *2015 xviii aisem annual conference*. IEEE. 2015, pp. 1–4.

[81] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. "Unsupervised visual domain adaptation using subspace alignment". In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 2960–2967.

[82] E. Fetaya and S. Ullman. "Learning local invariant mahalanobis distances". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 162–168.

[83] C. Francois. *Deep learning with Python*. 2017.

[84] E. R. Frederiks, K. Stenner, and E. V. Hobman. "The socio-demographic and psychological predictors of residential energy consumption: A comprehensive review". In: *Energies* 8.1 (2015), pp. 573–609.

[85] E. Frentzos, K. Gratsias, and Y. Theodoridis. "Index-based most similar trajectory search". In: *2007 IEEE 23rd International Conference on Data Engineering*. IEEE. 2007, pp. 816–825.

[86] Q. Fu, Q. Liu, Z. Gao, H. Wu, B. Fu, and J. Chen. "A Building Energy Consumption Prediction Method Based on Integration of a Deep Neural Network and Transfer Reinforcement Learning". In: *International Journal of Pattern Recognition and Artificial Intelligence* (2019).

[87] F. Fuerst, D. Kavarnou, R. Singh, and H. Adan. "Determinants of energy consumption and exposure to energy price risk: a UK study". In: *Zeitschrift für Immobilienökonomie* 6.1 (2020), pp. 65–80.

[88] *gbXML. Green Building XML*. Accessed: 2019-09-20.

[89] F. A. Gers, J. Schmidhuber, and F. Cummins. "Learning to forget: Continual prediction with LSTM". In: (1999).

[90] M. Ghifary, W. Bastiaan Kleijn, M. Zhang, and D. Balduzzi. “Domain generalization for object recognition with multi-task autoencoders”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2551–2559.

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

[92] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[93] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[94] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf. “Covariate shift by kernel mean matching”. In: *Dataset shift in machine learning* 3.4 (2009), p. 5.

[95] T. R. Gruber. “A translation approach to portable ontology specifications”. In: *Knowledge acquisition* 5.2 (1993), pp. 199–220.

[96] N. Guarino, D. Oberle, and S. Staab. “What is an ontology?” In: *Handbook on ontologies*. Springer, 2009, pp. 1–17.

[97] I. Guyon, A. Saffari, G. Dror, and G. Cawley. “Model Selection: Beyond the Bayesian/Frequentist Divide.” In: *Journal of Machine Learning Research* 11.1 (2010).

[98] R. Haas. “Energy efficiency indicators in the residential sector: What do we know and what has to be ensured?” In: *Energy Policy* 25.7-9 (1997), pp. 789–802.

[99] R. Hadsell, S. Chopra, and Y. LeCun. “Dimensionality reduction by learning an invariant mapping”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. Vol. 2. IEEE. 2006, pp. 1735–1742.

[100] F. Haller et al. “ARMILLA-ein Installationsmodell”. In: *Institut f ur Industrielle Bauproduktion, Universit at Karlsruhe* (1985).

[101] *Haystack Tagging Ontology*. https://lov.linkeddata.es/dataset/lov/vocabs/hto. Accessed: 2021-04-20.

[102] G. E. Hinton, S. Osindero, and Y.-W. Teh. “A fast learning algorithm for deep belief nets”. In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[103] G. E. Hinton and R. R. Salakhutdinov. “Reducing the dimensionality of data with neural networks”. In: *science* 313.5786 (2006), pp. 504–507.

[104] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[105] J. Hoffman, B. Kulis, T. Darrell, and K. Saenko. “Discovering latent domains for multisource domain adaptation”. In: *European Conference on Computer Vision*. Springer. 2012, pp. 702–715.

[106] T. Hofmann, B. Schölkopf, and A. J. Smola. “Kernel methods in machine learning”. In: *The annals of statistics* 36.3 (2008), pp. 1171–1220.

[107] A. Hooshmand and R. Sharma. “Energy Predictive Models with Limited Data using Transfer Learning”. In: *Proceedings of the Tenth ACM International Conference on Future Energy Systems*. 2019, pp. 12–16.

[108] K. Hornik. “Approximation capabilities of multilayer feedforward networks”. In: *Neural networks* 4.2 (1991), pp. 251–257.

[109] K. Hornik, M. Stinchcombe, H. White, et al. “Multilayer feedforward networks are universal approximators.” In: *Neural networks* 2.5 (1989), pp. 359–366.

[110] J. Hu, J. Lu, and Y.-P. Tan. “Discriminative deep metric learning for face verification in the wild”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1875–1882.

[111] 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*. 2007, pp. 601–608.

[112] D. H. Hubel and T. N. Wiesel. “Receptive fields and functional architecture of monkey striate cortex”. In: *The Journal of physiology* 195.1 (1968), pp. 215–243.

[113] F. Iglesias and W. Kastner. “Analysis of similarity measures in times series clustering for the discovery of building energy patterns”. In: *Energies* 6.2 (2013), pp. 579–597.

[114] H. Izakian, W. Pedrycz, and I. Jamal. “Fuzzy clustering of time series data using dynamic time warping distance”. In: *Engineering Applications of Artificial Intelligence* 39 (2015), pp. 235–244.

[115] A. K. Jain. “Data clustering: 50 years beyond K-means”. In: *Pattern recognition letters* 31.8 (2010), pp. 651–666.

[116] R. K. Jain, K. M. Smith, P. J. Culligan, and J. E. Taylor. “Forecasting energy consumption of multi-family residential buildings using support vector regression: Investigating the impact of temporal and spatial monitoring granularity on performance accuracy”. In: *Applied Energy* 123 (2014), pp. 168–178.

[117] N. Japkowicz and S. Stephen. “The class imbalance problem: A systematic study”. In: *Intelligent data analysis* 6.5 (2002), pp. 429–449.

[118] I.-H. Jhuo, D. Liu, D. Lee, and S.-F. Chang. “Robust visual domain adaptation with low-rank reconstruction”. In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 2168–2175.

[119] J. H. Kämpf and D. Robinson. “A simplified thermal model to support analysis of urban resource flows”. In: *Energy and buildings* 39.4 (2007), pp. 445–453.

[120] A. Karpathy. *Software 2.0*. Medium `https://medium.com/@karpathy/software-2-0-a64152b37c35`. Accessed: 2021-01-05. 2017.

[121] A. Karpathy. *The Unreasonable Effectiveness of Recurrent Neural Networks*. `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`. 2015.

[122] M. Kaya and H. Ş. Bilge. “Deep metric learning: A survey”. In: *Symmetry* 11.9 (2019), p. 1066.

[123] F. Kaytez, M. C. Taplamacioglu, E. Cam, and F. Hardalac. “Forecasting electricity consumption: A comparison of regression analysis, neural networks and least squares support vector machines”. In: *International Journal of Electrical Power & Energy Systems* 67 (2015), pp. 431–438.

[124] E. Keogh and C. A. Ratanamahatana. “Exact indexing of dynamic time warping”. In: *Knowledge and information systems* 7.3 (2005), pp. 358–386.

[125] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba. "Undoing the damage of dataset bias". In: *European Conference on Computer Vision*. Springer. 2012, pp. 158–171.

[126] E.-J. Kim, G. Plessis, J.-L. Hubert, and J.-J. Roux. "Urban energy simulation: Simplification and reduction of building envelope models". In: *Energy and Buildings* 84 (2014), pp. 193–202.

[127] T.-Y. Kim and S.-B. Cho. "Predicting residential energy consumption using CNN-LSTM neural networks". In: *Energy* 182 (2019), pp. 72–81.

[128] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[129] W. Kong, Z. Y. Dong, D. J. Hill, F. Luo, and Y. Xu. "Short-term residential load forecasting based on resident behaviour learning". In: *IEEE Transactions on Power Systems* 33.1 (2017), pp. 1087–1088.

[130] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang. "Short-term residential load forecasting based on LSTM recurrent neural network". In: *IEEE Transactions on Smart Grid* 10.1 (2017), pp. 841–851.

[131] W. M. Kouw and M. Loog. "A review of domain adaptation without target labels". In: *IEEE transactions on pattern analysis and machine intelligence* (2019).

[132] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[133] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[134] B. Kulis et al. "Metric learning: A survey". In: *Foundations and trends in machine learning* 5.4 (2012), pp. 287–364.

[135] B. Kumari and T. Swarnkar. "Filter versus wrapper feature subset selection in large dimensionality micro array: A review". In: (2011).

[136] T. Kusuda. "Fundamentals of building heat transfer". In: *Journal of Research of the National Bureau of Standards* 82.2 (1977), pp. 97–106.

[137] A. L'heureux, K. Grolinger, H. F. Elyamany, and M. A. Capretz. "Machine learning with big data: Challenges and approaches". In: *Ieee Access* 5 (2017), pp. 7776–7797.

[138] M. Labiadh, C. Obrecht, C. F. da Silva, and P. Ghodous. "A microservice-based framework for exploring data selection in cross-building knowledge transfer". In: *Service Oriented Computing and Applications* (2020), pp. 1–11.

[139] D. Lahat, T. Adali, and C. Jutten. "Multimodal data fusion: an overview of methods, challenges, and prospects". In: *Proceedings of the IEEE* 103.9 (2015), pp. 1449–1477.

[140] G. Lai, W.-C. Chang, Y. Yang, and H. Liu. "Modeling long-and short-term temporal patterns with deep neural networks". In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 2018, pp. 95–104.

[141] K. Lam, N. Wong, A. Mahdavi, K. Chan, Z. Kang, and S. Gupta. "SEMPER-II: an internet-based multi-domain building performance simulation environment for early design support". In: *Automation in Construction* 13.5 (2004), pp. 651–663.

[142] Y. LeCun, Y. Bengio, et al. "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.

[143] M. Lenzerini. "Data integration: A theoretical perspective". In: *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 2002, pp. 233–246.

[144] C. Li, Z. Ding, D. Zhao, J. Yi, and G. Zhang. "Building energy consumption prediction: An extreme deep learning approach". In: *Energies* 10.10 (2017), p. 1525.

[145] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. "Deeper, broader and artier domain generalization". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5542–5550.

[146] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. "Learning to generalize: Meta-learning for domain generalization". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[147] H. Li, S. Jialin Pan, S. Wang, and A. C. Kot. "Domain generalization with adversarial feature learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5400–5409.

[148] J. Li. "Assessing the accuracy of predictive models for numerical data: Not r nor r2, why not? Then what?" In: *PloS one* 12.8 (2017), e0183250.

[149] Q. Li, P. Ren, and Q. Meng. "Prediction model of annual energy consumption of residential buildings". In: *2010 international conference on advances in energy engineering*. IEEE. 2010, pp. 223–226.

[150] W. Li, Y. Zhou, K. Cetin, J. Eom, Y. Wang, G. Chen, and X. Zhang. "Modeling urban building energy use: A review of modeling approaches and procedures". In: *Energy* 141 (2017), pp. 2445–2457.

[151] Y. Li, X. Tian, M. Gong, Y. Liu, T. Liu, K. Zhang, and D. Tao. "Deep domain generalization via conditional invariant adversarial networks". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 624–639.

[152] Y. Li, Y. Yang, W. Zhou, and T. M. Hospedales. "Feature-critic networks for heterogeneous domain generalization". In: *arXiv preprint arXiv:1901.11448* (2019).

[153] B. Lim and S. Zohren. "Time Series Forecasting With Deep Learning: A Survey". In: *arXiv preprint arXiv:2004.13408* (2020).

[154] Z. Lipton, Y.-X. Wang, and A. Smola. "Detecting and correcting for label shift with black box predictors". In: *International conference on machine learning*. PMLR. 2018, pp. 3122–3130.

[155] Z. C. Lipton, J. Berkowitz, and C. Elkan. "A critical review of recurrent neural networks for sequence learning". In: *arXiv preprint arXiv:1506.00019* (2015).

[156] Z. C. Lipton. "A Critical Review of Recurrent Neural Networks for Sequence Learning". In: *CoRR* abs/1506.00019 (2015). URL: http://arxiv.org/abs/1506.00019.

[157] Y. Liu, M. C. Roberts, and R. Sioshansi. "A vector autoregression weather model for electricity supply and demand modeling". In: *Journal of Modern Power Systems and Clean Energy* 6.4 (2018), pp. 763–776.

[158] M. Long, Y. Cao, J. Wang, and M. Jordan. "Learning transferable features with deep adaptation networks". In: *International conference on machine learning*. PMLR. 2015, pp. 97–105.

[159] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang. "Learning under concept drift: A review". In: *IEEE Transactions on Knowledge and Data Engineering* 31.12 (2018), pp. 2346–2363.

[160] J. Lu, J. Hu, and J. Zhou. "Deep metric learning for visual understanding: An overview of recent advances". In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 76–84.

[161] Z. Ma and L. Yan. *Emerging Technologies and Applications in Data Processing and Management*. IGI Global., 2019.

[162] M. Mancini, S. R. Bulo, B. Caputo, and E. Ricci. "Adagraph: Unifying predictive and continuous domain adaptation through graphs". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6568–6577.

[163] M. Mancini, S. R. Bulò, B. Caputo, and E. Ricci. "Best sources forward: domain generalization through source-specific nets". In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE. 2018, pp. 1353–1357.

[164] M. Mancini, L. Porzi, S. Rota Bulò, B. Caputo, and E. Ricci. "Boosting domain adaptation by discovering latent domains". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3771–3780.

[165] Y. Mansour, M. Mohri, and A. Rostamizadeh. "Domain adaptation with multiple sources". In: *Advances in neural information processing systems* 21 (2008), pp. 1041–1048.

[166] Y. Mansour, M. Mohri, and A. Rostamizadeh. "Domain adaptation: Learning bounds and algorithms". In: *arXiv preprint arXiv:0902.3430* (2009).

[167] W. Mao, M. Tian, and G. Yan. "Research of load identification based on multiple-input multiple-output SVM model selection". In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 226.5 (2012), pp. 1395–1409.

[168] D. L. Marino, K. Amarasinghe, and M. Manic. "Building energy load forecasting using deep neural networks". In: *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*. IEEE. 2016, pp. 7046–7051.

[169] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker,

Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[170] P. A. Mathew, L. N. Dunn, M. D. Sohn, A. Mercado, C. Custudio, and T. Walter. “Big-data for building energy performance: Lessons from assembling a very large national database of building energy use”. In: *Applied Energy* 140 (2015), pp. 85–93.

[171] W. S. McCulloch and W. Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[172] I. Melekhov, J. Kannala, and E. Rahtu. “Siamese network features for image matching”. In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE. 2016, pp. 378–383.

[173] M. Minsky and S. A. Papert. *Perceptrons: An introduction to computational geometry*. MIT press, 2017.

[174] T. M. Mitchell et al. “Machine learning. 1997”. In: *Burr Ridge, IL: McGraw Hill* 45.37 (1997), pp. 870–877.

[175] E. Mocanu, P. H. Nguyen, M. Gibescu, and W. L. Kling. “Deep learning for estimating building energy consumption”. In: *Sustainable Energy, Grids and Networks* 6 (2016), pp. 91–99.

[176] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[177] C. Moreira. “Learning to rank academic experts”. PhD thesis. Master Thesis, Technical University of Lisbon, 2011.

[178] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto. “Unified deep supervised domain adaptation and generalization”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5715–5725.

[179] K. Muandet, D. Balduzzi, and B. Schölkopf. “Domain generalization via invariant feature representation”. In: *International Conference on Machine Learning*. 2013, pp. 10–18.

[180] M. Müller. “Dynamic time warping”. In: *Information retrieval for music and motion* (2007), pp. 69–84.

[181] D. Murray, L. Stankovic, and V. Stankovic. “An electrical load measurements dataset of United Kingdom households from a two-year longitudinal study”. In: *Scientific data* 4 (2017), p. 160122.

[182] B. Nepal, M. Yamaha, A. Yokoe, and T. Yamaji. “Electricity load forecasting using clustering and ARIMA model for energy management in buildings”. In: *Japan Architectural Review* 3.1 (2020), pp. 62–76.

[183] S. Newman. *Building microservices: designing fine-grained systems*. " O’Reilly Media, Inc.", 2015.

[184] G. R. Newsham and B. J. Birt. “Building-level occupancy data to improve ARIMA-based electricity use forecasts”. In: *Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building*. 2010, pp. 13–18.

[185] NIBS. *United States national building information modeling standard version 1—Part 1: Overview, principles, and methodologies*. 2008.

[186] S. Nowozin, B. Cseke, and R. Tomioka. "f-gan: Training generative neural samplers using variational divergence minimization". In: *Advances in neural information processing systems*. 2016, pp. 271–279.

[187] 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". In: *arXiv preprint arXiv:1609.03499* (2016).

[188] E. Ostertagová. "Modelling using polynomial regression". In: *Procedia Engineering* 48 (2012), pp. 500–506.

[189] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. "Domain adaptation via transfer component analysis". In: *IEEE Transactions on Neural Networks* 22.2 (2011), pp. 199–210.

[190] S. J. Pan, Q. Yang, et al. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2010), pp. 1345–1359.

[191] R. Pascanu, T. Mikolov, and Y. Bengio. "On the Difficulty of Training Recurrent Neural Networks". In: *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. ICML'13. Atlanta, GA, USA: JMLR.org, 2013, pp. III-1310–III-1318. URL: http://dl.acm.org/citation.cfm?id=3042817.3043083.

[192] R. Pascanu, T. Mikolov, and Y. Bengio. "On the difficulty of training recurrent neural networks". In: *International conference on machine learning*. 2013, pp. 1310–1318.

[193] P. Pauwels and W. Terkaj. "EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology". In: *Automation in Construction* 63 (2016), pp. 100–133.

[194] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[195] P. O. Pinheiro. "Unsupervised domain adaptation with similarity learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8004–8013.

[196] J. Pohl, L. Myers, J. Cotton, A. Chapman, J. Snyder, H. Chauvet, K. Pohl, and J. La Porta. "A computer-based design environment: implemented and planned extensions of the ICADS model". In: *Technicol Report, CADRU-06-92, CAD Research Center, Design and Construction Institute, College of Architecture and Environmental Design, Cal Poly, San Luis Obispo, California* (1992).

[197] M. Poveda-Villalón and R. G. Castro. *SAREF extension for building devices*. http://ontoology.linkeddata.es/publish/saref4bldg/index-en.html. Accessed: 2019-06-15. 2017.

[198] M. J. Pratt. "Introduction to ISO 10303—the STEP standard for product data exchange". In: *Journal of Computing and Information Science in Engineering* 1.1 (2001), pp. 102–103.

[199] M. Pritoni, D. Paine, G. Fierro, C. Mosiman, M. Poplawski, A. Saha, J. Bender, and J. Granderson. "Metadata Schemas and Ontologies for Building Energy Applications: A Critical Review and Use Case Analysis". In: *Energies* 14.7 (2021), p. 2024.

[200] *Project Haystack*. Accessed: 2019-10-10.

[201] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009. ISBN: 0262170051, 9780262170055.

[202] P. Ranacher and K. Tzavella. "How to compare movement? A review of physical movement similarity measures in geographic information science and beyond". In: *Cartography and geographic information science* 41.3 (2014), pp. 286–307.

[203] S. Raschka. "Model evaluation, model selection, and algorithm selection in machine learning". In: *arXiv preprint arXiv:1811.12808* (2018).

[204] M. H. Rasmussen, P. Pauwels, C. A. Hviid, and J. Karlshøj. "Proposing a central AEC ontology that allows for domain specific extensions". In: *2017 Lean and Computing in Construction Congress*. 2017.

[205] C. Reinisch, M. Kofler, F. Iglesias, and W. Kastner. "Thinkhome energy efficiency in future smart homes". In: *EURASIP Journal on Embedded Systems* 2011.1 (2011), p. 104617.

[206] M. Ribeiro, K. Grolinger, H. F. ElYamany, W. A. Higashino, and M. A. Capretz. "Transfer learning with seasonal and trend adjustment for cross-building energy forecasting". In: *Energy and Buildings* 165 (2018), pp. 352–363.

[207] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro. "Learning to learn without forgetting by maximizing transfer and minimizing interference". In: *arXiv preprint arXiv:1810.11910* (2018).

[208] Y. Roh, G. Heo, and S. E. Whang. "A Survey on Data Collection for Machine Learning: a Big Data-AI Integration Perspective". In: *arXiv preprint arXiv:1811.03402* (2018).

[209] F. Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[210] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich. "To transfer or not to transfer". In: *NIPS 2005 workshop on transfer learning*. Vol. 898. 2005, pp. 1–4.

[211] R. Rosipal, M. Girolami, L. J. Trejo, and A. Cichocki. "Kernel PCA for feature extraction and de-noising in nonlinear regression". In: *Neural Computing & Applications* 10.3 (2001), pp. 231–243.

[212] S. K. Roy, M. Harandi, R. Nock, and R. Hartley. "Siamese networks: The tale of two manifolds". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 3046–3055.

[213] A. Rozantsev, M. Salzmann, and P. Fua. "Beyond sharing weights for deep domain adaptation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018).

[214] S. Ruder, P. Ghaffari, and J. G. Breslin. "Data selection strategies for multi-domain sentiment analysis". In: *arXiv preprint arXiv:1702.02426* (2017).

[215] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1". In: ed. by D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group. Cambridge, MA, USA: MIT Press, 1986. Chap. Learning Internal Representations by Error Propagation, pp. 318–362. ISBN: 0-262-68053-X. URL: http://dl.acm.org/citation.cfm?id=104279.104293.

[216] J. Runge and R. Zmeureanu. "Forecasting energy use in buildings using artificial neural networks: a review". In: *Energies* 12.17 (2019), p. 3254.

[217] J. H. Rutherford. "KNODES: knowledge-based design decision support". In: *Proceedings of the fifth international conference on Computer-aided architectural design futures*. North-Holland Publishing Co. 1993, pp. 357–374.

[218] E. M. Ryan and T. F. Sanquist. "Validation of building energy modeling tools under idealized and realistic conditions". In: *Energy and Buildings* 47 (2012), pp. 375–382.

[219] H. Sakoe and S. Chiba. "Dynamic programming algorithm optimization for spoken word recognition". In: *IEEE transactions on acoustics, speech, and signal processing* 26.1 (1978), pp. 43–49.

[220] S. Salvador and P. Chan. "Toward accurate dynamic time warping in linear time and space". In: *Intelligent Data Analysis* 11.5 (2007), pp. 561–580.

[221] M. Sánchez-Fernández, M. de-Prado-Cumplido, J. Arenas-Garcıa, and F. Pérez-Cruz. "SVM multiregression for nonlinear channel estimation in multiple-input multiple-output systems". In: *IEEE transactions on signal processing* 52.8 (2004), pp. 2298–2307.

[222] C. Sandels, D. Brodén, J. Widén, L. Nordström, and E. Andersson. "Modeling office building consumer load with a combined physical and behavioral approach: Simulation and validation". In: *Applied energy* 162 (2016), pp. 472–485.

[223] S. Sayad. *Support Vector Machine - Regression*. https://www.saedsayad.com/support_vector_machine_reg.htm. Accessed: 2020-10-01.

[224] J. Serra and J. L. Arcos. "An empirical evaluation of similarity measures for time series classification". In: *Knowledge-Based Systems* 67 (2014), pp. 305–314.

[225] S. Seyedzadeh, F. P. Rahimian, I. Glesk, and M. Roper. "Machine learning for estimation of building energy consumption and performance: a review". In: *Visualization in Engineering* 6.1 (2018), p. 5.

[226] J. Shen, Y. Qu, W. Zhang, and Y. Yu. "Wasserstein distance guided representation learning for domain adaptation". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[227] D. F. Silva and G. E. Batista. “Speeding up all-pairwise dynamic time warping matrix calculation”. In: *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM. 2016, pp. 837–845.

[228] A. J. Smola and B. Schölkopf. “A tutorial on support vector regression”. In: *Statistics and computing* 14.3 (2004), pp. 199–222.

[229] D. M. Solomon, R. L. Winter, A. G. Boulanger, R. N. Anderson, and L. L. Wu. “Forecasting energy demand in large commercial buildings using support vector machine regression”. In: (2011).

[230] A. Sorjamaa and A. Lendasse. “Time series prediction using DirRec strategy.” In: *Esann*. Vol. 6. Citeseer. 2006, pp. 143–148.

[231] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[232] T. G. Stavropoulos, D. Vrakas, D. Vlachava, and N. Bassiliades. “BOnSAI: a smart building ontology for ambient intelligence”. In: *Proceedings of the 2nd international conference on web intelligence, mining and semantics*. 2012, pp. 1–12.

[233] R. Studer, V. R. Benjamins, and D. Fensel. “Knowledge engineering: Principles and methods”. In: *Data & knowledge engineering* 25.1-2 (1998), pp. 161–197.

[234] M. Sugiyama, S. Nakajima, H. Kashima, P. V. Buenau, and M. Kawanabe. “Direct importance estimation with model selection and its application to covariate shift adaptation”. In: *Advances in neural information processing systems*. 2008, pp. 1433–1440.

[235] M. Sugiyama and A. J. Storkey. “Mixture regression for covariate shift”. In: *Advances in Neural Information Processing Systems*. 2007, pp. 1337–1344.

[236] Q. Sun, R. Chattopadhyay, S. Panchanathan, and J. Ye. “A two-stage weighting framework for multi-source domain adaptation”. In: *Advances in neural information processing systems*. 2011, pp. 505–513.

[237] I. Sutskever. “Training Recurrent Neural Networks - Ilia Sutskever - PhD thesis”. In: (). URL: `http://www.cs.utoronto.ca/%5C%7Eilya/pubs/ilya%5C_sutskever%5C_phd%5C_thesis.pdf`.

[238] I. Sutskever, O. Vinyals, and Q. V. Le. “Sequence to sequence learning with neural networks”. In: *arXiv preprint arXiv:1409.3215* (2014).

[239] D. Taibi, V. Lenarduzzi, and C. Pahl. “Microservices anti-patterns: A taxonomy”. In: *Microservices*. Springer, 2020, pp. 111–128.

[240] S. B. Taieb, G. Bontempi, A. Sorjamaa, and A. Lendasse. “Long-term prediction of time series by combining direct and mimo strategies”. In: *2009 International Joint Conference on Neural Networks*. IEEE. 2009, pp. 3054–3061.

[241] Y. Taigman, A. Polyak, and L. Wolf. “Unsupervised cross-domain image generation”. In: *arXiv preprint arXiv:1611.02200* (2016).

[242] B. Tan, Y. Zhang, S. J. Pan, and Q. Yang. “Distant domain transfer learning”. In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[243] V. Tatsiankou. "Instrumentation development for site-specific prediction of spectral effects on concentrated photovoltaic system performance". PhD thesis. Université d'Ottawa/University of Ottawa, 2014.

[244] The Energy Systems Research Unit (ESRU). *ESP-r*. `https://www.strath.ac.uk/research/energysystemsresearchunit/applications/esp-r/`. Accessed: 2020-10-01. 2011.

[245] C. Tian, J. Ma, C. Zhang, and P. Zhan. "A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network". In: *Energies* 11.12 (2018), p. 3493.

[246] K. M. Ting and I. H. Witten. "Stacked Generalization: when does it work?" In: (1997).

[247] University of Wisconsin-Madison. *A TRaNsient SYstems Simulation Program*. `https://sel.me.wisc.edu/trnsys/`. Accessed: 2021-06-10. 2015.

[248] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. "Attention is all you need". In: *arXiv preprint arXiv:1706.03762* (2017).

[249] M. Vlachos, G. Kollios, and D. Gunopulos. "Discovering similar multidimensional trajectories". In: *Proceedings 18th international conference on data engineering*. IEEE. 2002, pp. 673–684.

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

[251] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. "Experimental comparison of representation methods and distance measures for time series data". In: *Data Mining and Knowledge Discovery* 26.2 (2013), pp. 275–309.

[252] Y. Wang, M. Liu, Z. Bao, and S. Zhang. "Short-term load forecasting with multi-source data using gated recurrent unit neural networks". In: *Energies* 11.5 (2018), p. 1138.

[253] Z. Wang and R. S. Srinivasan. "A review of artificial intelligence based building energy prediction with a focus on ensemble prediction models". In: *2015 Winter Simulation Conference (WSC)*. IEEE. 2015, pp. 3438–3448.

[254] Z. Wang, M. Loog, and J. van Gemert. "Respecting Domain Relations: Hypothesis Invariance for Domain Generalization". In: *arXiv preprint arXiv:2010.07591* (2020).

[255] 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*. 2019, pp. 11293–11302.

[256] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean. "Characterizing concept drift". In: *Data Mining and Knowledge Discovery* 30.4 (2016), pp. 964–994.

[257] P. J. WERBOS. "Backpropagation Through Time: What It Does and How to Do It". In: 2004.

[258] R. J. Williams and D. Zipser. "A learning algorithm for continually running fully recurrent neural networks". In: *Neural computation* 1.2 (1989), pp. 270–280.

[259] R. J. Williams and J. Peng. "An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories". In: *Neural Computation* 2 (1990), pp. 490–501.

[260] G. Wilson and D. J. Cook. "A survey of unsupervised deep domain adaptation". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 11.5 (2020), pp. 1–46.

[261] P. Wilson and P. Kennicott. "Iso step baseline requirements document (ipim)". In: *Technical report, ISO TC 184/SC4/WG1 Document Number N284* (1988).

[262] D. H. Wolpert. "Stacked generalization". In: *Neural networks* 5.2 (1992), pp. 241–259.

[263] D. H. Wolpert. "The lack of a priori distinctions between learning algorithms". In: *Neural computation* 8.7 (1996), pp. 1341–1390.

[264] C. Won, S. No, and Q. Alhadidi. "Factors affecting energy performance of large-scale office buildings: Analysis of benchmarking data from New York City and Chicago". In: *Energies* 12.24 (2019), p. 4783.

[265] C. Xiong, S. McCloskey, S.-H. Hsieh, and J. J. Corso. "Latent domains modeling for visual domain adaptation". In: *Twenty-Eighth AAAI Conference on Artificial Intelligence*. 2014.

[266] H. Xu, J. Zheng, and R. Chellappa. "Bridging the Domain Shift by Domain Adaptive Dictionary Learning." In: *BMVC*. 2015, pp. 96–1.

[267] R. Xu, Z. Chen, W. Zuo, J. Yan, and L. Lin. "Deep cocktail network: Multi-source unsupervised domain adaptation with category shift". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3964–3973.

[268] Z. Xu, W. Li, L. Niu, and D. Xu. "Exploiting low-rank structure from latent domains for domain generalization". In: *European Conference on Computer Vision*. Springer. 2014, pp. 628–643.

[269] K. Yan, X. Wang, Y. Du, N. Jin, H. Huang, and H. Zhou. "Multi-step short-term power consumption forecasting with a hybrid deep learning strategy". In: *Energies* 11.11 (2018), p. 3089.

[270] J. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy. "Deep convolutional neural networks on multichannel time series for human activity recognition". In: *Twenty-fourth international joint conference on artificial intelligence*. 2015.

[271] 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*. ACM. 2007, pp. 188–197.

[272] Y. Yang and T. M. Hospedales. "A unified perspective on multi-domain and multi-task learning". In: *arXiv preprint arXiv:1412.7489* (2014).

[273] Y. Yang and T. M. Hospedales. "Multivariate regression on the grassmannian for predicting novel domains". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 5071–5080.

[274] F. Yu and V. Koltun. "Multi-scale context aggregation by dilated convolutions". In: *arXiv preprint arXiv:1511.07122* (2015).

[275] Z. Yu, F. Haghighat, B. C. Fung, and H. Yoshino. "A decision tree method for building energy demand modeling". In: *Energy and Buildings* 42.10 (2010), pp. 1637–1646.

[276] B. Zadrozny. "Learning and evaluating classifiers under sample selection bias". In: *Proceedings of the twenty-first international conference on Machine learning*. ACM. 2004, p. 114.

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

[278] L. Zhang. "Transfer adaptation learning: A decade survey". In: *arXiv preprint arXiv:1903.04687* (2019).

[279] L. Zhang, J. Fu, S. Wang, D. Zhang, Z. Dong, and C. P. Chen. "Guide subspace learning for unsupervised domain adaptation". In: *IEEE transactions on neural networks and learning systems* 31.9 (2019), pp. 3374–3388.

[280] L. Zhang, W.-D. Zhou, P.-C. Chang, J.-W. Yang, and F.-Z. Li. "Iterated time series prediction with multiple support vector regression models". In: *Neurocomputing* 99 (2013), pp. 411–422.

[281] H. X. Zhao and F. Magoulès. "Parallel support vector machines applied to the prediction of multiple buildings energy consumption". In: *Journal of Algorithms & Computational Technology* 4.2 (2010), pp. 231–249.

[282] H.-x. Zhao and F. Magoulès. "A review on the prediction of building energy consumption". In: *Renewable and Sustainable Energy Reviews* 16.6 (2012), pp. 3586–3592.

[283] S. Zhao, B. Li, P. Xu, and K. Keutzer. "Multi-source Domain Adaptation in the Deep Learning Era: A Systematic Survey". In: *arXiv preprint arXiv:2002.12169* (2020).

[284] J. Zheng, C. Xu, Z. Zhang, and X. Li. "Electric load forecasting in smart grids using long-short-term-memory based recurrent neural network". In: *2017 51st Annual Conference on Information Sciences and Systems (CISS)*. IEEE. 2017, pp. 1–6.

[285] 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*. Springer. 2014, pp. 298–310.

[286] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy. "Domain generalization: A survey". In: *arXiv preprint arXiv:2103.02503* (2021).

[287] Y. Zhu, F. Zhuang, and D. Wang. "Aligning domain-specific distribution and classifier for cross-domain classification from multiple sources". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 5989–5996.

[288] F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He. "Supervised representation learning: Transfer learning with deep autoencoders". In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*. 2015.