Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN

COMPUTER SCIENCE AND ENGINEERING

Ciclo 36

Settore Concorsuale: 09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Settore Scientifico Disciplinare: ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

INFORMED MACHINE LEARNING FOR EPIDEMICS: FROM DATA ANALYSIS TO TIME-SERIES FORECASTING

Presentata da: Federico Baldo

Coordinatore Dottorato	**Supervisore**
Ilaria Bartolini	Michela Milano

Esame finale anno 2024

Abstract

This research dives deep into the application of Machine Learning (ML) in epidemiology, aiming to craft innovative strategies to address challenges presented by epidemic phenomena. Recognizing the dual nature of epidemics — as complex physical occurrences and as catalysts of pressure on healthcare systems — we focus on two main areas: data analysis for patient condition assessment and time-series forecasting for epidemiological trend prediction.

In the realm of data analysis, we introduced a tree-based ML pipeline apt at evaluating the severity of infections using patient records. On the forecasting front, we addressed the challenges posed by the scarcity of data and the intricate nature of epidemics, developing methodologies for predicting epidemiological trends. In this setting, we exploit domain knowledge to guide the learning process, ensuring accurate and reliable predictions even when confronted with incomplete and noisy data.

Our contributions include an extensive review of deep learning models incorporating constraint-based domain knowledge to enhance performance. Additionally, we provide a thorough overview of deep learning applications in epidemic forecasting, highlighting a wide set of approaches and their effectiveness. Concerning data analysis, we propose a decision tree-based approach for assessing Sars-CoV-2 patient risk, alongside the deployment of a data augmentation technique and a decorrelation method to enhance the robustness and reliability of the analysis. In the realm of time-series forecasting, we conducted a comprehensive exploration of Physics Informed Neural Networks and their applications in modeling epidemic phenomena. Furthermore, we introduced a novel methodology that integrates deep neural networks with renewal processes for time-series forecasting. Both these methods are further enhanced with the addition of transfer learning, which allows the approximation of an accurate predictor with scarce data.

The approaches presented exhibit enhanced performance compared to other deep learning methods, marking a promising avenue for future research. Indeed, this research showcases some of the potential applications of ML in epidemiology, motivating future extensions and the significance of interdisciplinary approaches in addressing the complex challenges presented by epidemics.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The 21st century has been marked by an ever-increasing level of human activity on the planet, favoring a global interconnection and new development opportunities. Unfortunately, this phenomenon has led to environment-invasive operations that altered the equilibrium of our ecosystem, paving the way for the emergence of novel viruses [12] (Figure 1.1). Among these, *zoonotic* viruses are the most worrying, due to their unpredictability and their capability of spreading at a fast pace (e.g., SARS, MERS, H1N1). Most notable, is the recent outbreak of Sars-CoV-2 [154], which started in late 2019, in Wuhan, China. This unexpected scenario has brought to light the unpreparedness of policy-makers and the consequent lack of prompt and effective responses apt at containing the spread of the virus. A key element to counteract this scenario is the availability of preemptive mechanisms capable of assessing the impact of specific decisions, or policies, on the evolution of the outbreak. Such mechanisms should be capable of coping with the complexity and uncertainty aspects ingrained in the epidemic phenomena, which are hard to solve with traditional approaches from the statistical and epidemiological fields - for instance, analytically expressing the relationship between virus spreading, countermeasures, and socio-economic impacts is a non-trivial task. The development of new technologies providing the aforementioned capabilities is of crucial relevance to mitigating the negative effects of epidemics. Artificial Intelligence (AI), and more specifically Machine Learning (ML), are emerging as powerful tools in this arena, showing promising results in addressing a broad spectrum of challenges. This is particularly evident in their application to healthcare and public health issues [129]. The field of epidemiology has also witnessed a surge in ML-driven research, particularly in the wake of the COVID-19 pandemic, underscoring the potential of these methodologies in contributing to our readiness and response strategies

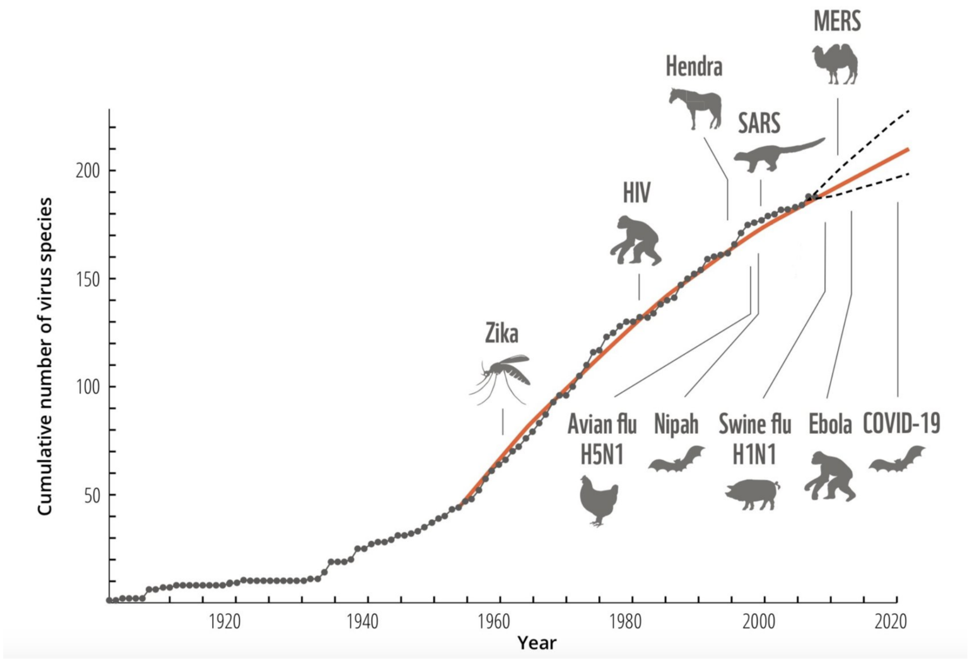


Figure 1.1: Cumulative number of zoonotic viruses in the last +100 years - `source: WWF Science Global 2020`

against future global outbreaks.

1.1 Contribution

The objective of this research was to design, develop, and apply novel approaches based on Machine Learning methodologies to tackle epidemiological problems. In doing so we identified two critical areas in which our contribution could be significant. Indeed, epidemics have a duplicitous nature: on the one hand, they are complex physical phenomena, which are very hard to capture and describe; on the other hand, they affect people creating pressure on health care systems. Thus, our contribution was aimed at:

- Provide an analytical method to be used by physicians and researchers to assess the severity of an infection. In this sense, we developed a *data analysis* pipeline, which can be applied in a variety of contexts in the presence of tabular data representing the conditions of a patient. This project in particular required a close collaboration with health institutions, namely, the National Institute of Infectious Diseases "Lazzaro Spallazani" and the hospital "Policlinico Sant'Orsola-Malpighi" of

Bologna. Both these institutions provided guidance and validation for the solution provided.

- Development of methodologies forecasting epidemiological trends. This task represents a great challenge due to the lack of data and the complexity of the phenomenon. Even in this case, it was required the presence of domain experts guiding the development of the approaches, resulted in a visiting period at the Pierre Louis Institute of Epidemiology and Public Health in Paris.

The contributions of this work are the following:

- **Extensive Review of Deep Learning Models Enhanced by Constraint Based Domain Knowledge**: We conducted an in-depth review, focusing on improving deep learning models by leveraging constraint-based domain knowledge, thus enhancing their overall performance [20].

- **Comprehensive Overview of Deep Learning Methods for Epidemic Forecasting**: We provided a detailed survey of deep learning methods applied to the forecasting of epidemics. This overview summarizes various approaches and their efficacy in predicting epidemic trends [9].

- **Decision Tree-Based Data Analysis**: We introduced an innovative decision tree-based data analysis approach aimed at assessing factors associated with Sars-CoV-2 patients at higher risk of developing a severe form of illness. To achieve this, we employed data augmentation techniques and a decorrelation method to enhance the robustness and reliability of the analysis [10].

- **In-Depth Analysis of Universal Differential Equations**: Our research delves deeply into Universal Differential Equations, a machine learning-based method for approximating dynamic systems in a data-driven fashion. We focused on their applications in the context of epidemics [8, 145].

- **New Methodology Combining Deep Neural Networks and Renewal Processes for Time-Series Forecasting**: We introduced a novel methodology that integrates deep neural networks with renewal processes for time-series forecasting, specifically focusing on forecasting epidemic trends. Furthermore, this method is paired with a transfer learning mechanism to provide a more robust predictor.

Chapter 2

Background

2.1 Machine Learning

In recent decades, the increased computational capacity and the abundance of diverse datasets have paved the way for the widespread adoption of Machine Learning. This branch of AI aims to approximate functions in a data-driven fashion, allowing for the generalization of phenomena beyond the observed data samples.
To be more specific, the typical scenario involves having a training set denoted as D_{train}, which is used to construct the ML-based model. Additionally, we usually are provided with a test set, referred to as D_{test}, which is typically unknown at training time but is employed to assess the generalization capabilities of the approximated function. To this end, ML employs stochastic techniques apt at learning the statistical distribution underlying an observable event. ML can take different forms depending on the availability of data and the learning task. In this context, we can identify three broad categories:

- **Supervised Learning**. In this paradigm, ML adopts a mentor-student relationship, where the learning process revolves around a set of labeled inputs. Each input instance is coupled with an associated output. The ML model undergoes training to minimize the disparity between the input data and their corresponding labels.
- **Unsupervised Learning**. Unlike supervised learning, unsupervised learning grapples with unlabeled data. Here, the learning process attempts to uncover data patterns without explicit guidance, encompassing tasks like clustering and density estimation.

- **Reinforcement Learning**. This method finds application in scenarios involving agents acting within an environment. The objective is to identify a strategy that maps actions to environmental observations that maximize a predefined reward function.

In the context of this work, our primary focus will be on supervised learning. Therefore, when referring to ML, we implicitly allude to this particular approach. In other words, we can summarise an ML task as follows:

Provided a collection of variables X and a dependent variable y, interconnected through the function $f(X) = y$, our goal is to approximate a function $g(X)$, such that $g(X) \approx f(X)$

Depending on the domain of the predictive target, y, we define different learning tasks, namely:

- if y takes values in a finite set of discrete values we talk about a **classification task**, where each value in the domain represents a specific category associated with the input
- if y is continuous we talk about a **regression task**, meaning the outcome takes values in an infinite set of values.

One of the key ingredients of an ML model is the algorithm used to approximate the f function. Among the most popular approaches, we have Neural Networks, Support Vector Machines, and Decision Trees. The rationale behind the preference for ML-based methodologies over traditional techniques like Linear Regression and Logistic Regression lies in their capacity to capture intricate, non-linear correlations present in the data. This capability often unveils hidden insights that may remain concealed from human analysts or conventional statistical methodologies.

2.1.1 Decision Tree

Decision Trees are a Machine Learning methodology characterized by their tree-like structure. This structural aspect equips these models with a higher degree of interpretability. Specifically, Decision Trees provide a clear and direct graphical representation, facilitating a more intuitive understanding of their decision-making process [84]. The tree structure consists of three main components:

- **Nodes**. Nodes perform tests on specific attributes of the input instances.

- **Branches (or Splits)**. Branches, contingent on the test outcomes, direct the computation to the next node in the tree.
- **Leaf**. Leaves are terminal nodes labeled with the predicted outcome for the input instance.

The underlying concept is that X is composed of a set of attributes, $X = (a_1, \dots, a_n)$, that take different values depending on their domain; given a range of values admissible, we are going to be routed on different paths within the tree. The model stops when it reaches a leaf, which provides the final prediction.
The construction of a decision tree is incremental, with attributes progressively chosen for testing, leading to the creation of the next split in the tree. The process begins with a root node, always serving as the entry point for evaluating inputs. Attributes that yield the highest **Information Gain** for the predictive target are selected based on the training data. Information Gain helps identify attributes that can best organize the data given their associated outputs. Information Gain is computed using an ordering function, such as the Gini Index or Entropy. For example, in the case of Entropy, which measures disorder in a system (e.g., a dataset), the formula is:

$$E(S) = -\sum_c p_c \, log(p_c) \quad with \;\; p_c = \frac{|S_c|}{|S|}$$

Here, c represents a class, and S_c is the subset of training instances belonging to that class. Information Gain quantifies the reduction in Entropy achieved by partitioning the data using an attribute and its values:

$$I(S,a) = E(S) - \sum_{v \in V(a)} \frac{|S_{a=v}|}{|S|} E(S_{a=v})$$

Here, $V(a)$ represents the domain of attribute a. The algorithm progressively selects the attribute that yields the highest Information Gain, partitioning the training data into smaller subsets. When a partition contains only instances from the same class, a leaf node is added, labeled with the majority class. This method implicitly defines a metric of importance for features in the dataset. Information Gain helps quantify the relevance of an attribute with respect to the predictive target, providing a ranking of the most important attributes. This concept is referred to as Feature Importance and is a valuable tool for data analysis.
To illustrate how Decision Trees work and how explainable they are, we will provide a small example based on a toy problem. Below is a description of a dataset containing information about a group of patients, including:

- BMI, or Body Mass Index, which measures the amount of fat in an individual given height and weight. A BMI above 30 indicates obesity.
- Diabetes, a binary variable, which is going to be 1 if the patient is diabetic, 0 otherwise
- Smoker, a binary variable, which is going to be 1 if the patient is a regular smoker, 0 otherwise
- Heart Condition, a binary variable, which is going to be 1 if the patient has a heart condition, 0 otherwise

The primary objective is to predict whether a patient is likely to have a heart condition based on the information available in these variables. The samples representing our training instances are reported in the following table:

BMI	Diabetes	Smoker	Heart Condition
40	1	1	1
20	0	0	0
15	1	0	0
50	0	1	1
35	0	0	1
10	0	1	0
40	0	0	0

Table 2.1: Toy Dataset Heart Condition

The resulting Decision Tree is depicted in Figure 2.1, with the corresponding feature importance illustrated in Figure 3.3. Notably, the Decision Tree analysis reveals significant insights into the predictive factors. Observing the tree structure, it becomes evident that BMI plays a pivotal role in predicting the outcome. This significance is substantiated by the splitting value in the decision tree, which aligns closely with the obesity threshold, suggesting a strong correlation between BMI and the outcome. Furthermore, the analysis highlights the relevance of being a smoker as a factor contributing to predictions. On the other hand, the feature related to diabetes appears to hold little importance in predicting the outcome, as indicated by its minimal contribution to the Decision Tree structure. It's crucial to emphasize that the measure of feature importance does not imply causality. In other words, an important feature does not necessarily represent a causal factor for the outcome. Instead, we can speak of correlation, indicating that the variables are

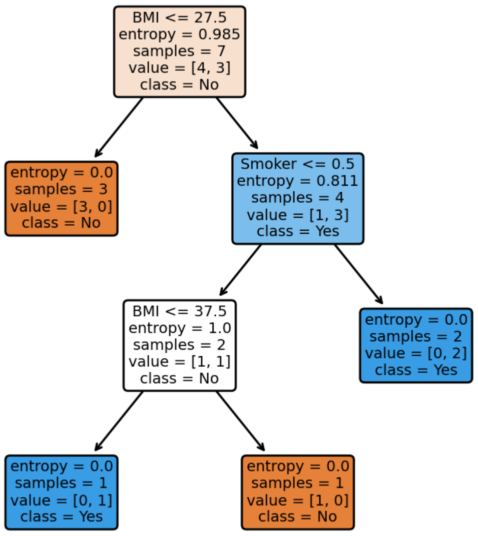


Figure 2.1: Decision Tree Example

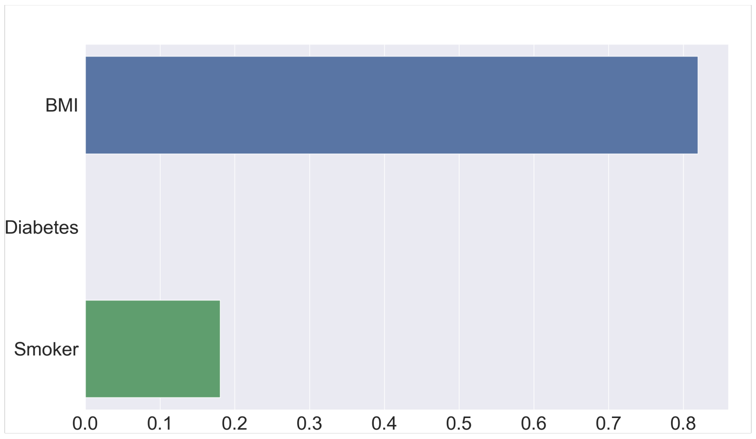


Figure 2.2: Feature Importance Example

related in some way. However, without further analysis, we cannot deduce the nature or direction of this relationship.

While Decision Trees are effective models for many tasks, they, like other machine learning models, have inherent limitations that are often encountered. Our primary goal in machine learning is to generalize the target function beyond the training sample. However, one common challenge is the issue of *overfitting*, which occurs when a model becomes too specific to the training data. Unfortunately, Decision Trees are susceptible to overfitting due to their reliance on the training sample. Indeed, they tend to create different models with minor variations in the data, which can lead to the model capturing noise or irregularities in the training set rather than the underlying patterns that generalize for unseen data (e.g., the test set). Overfitted Decision Trees may perform excellently on the training data but fail to generalize effectively to new data.

Random Forest

A widely embraced solution to combat the challenge of overfitting in Decision Trees is the application of a statistical technique known as *Bagging*. This method seeks to reduce the variance of the approximated function by training multiple Decision Trees on random subsets of variables, collectively forming what is referred to as a *Forest* [84]. Random Forests (RF) employ several strategies to enhance the limitations of individual Decision Trees: Firstly, instead of relying on a single Decision Tree, a Random Forest creates an ensemble of multiple trees, each trained on a different random subset of the available variables. For instance, following the example in the previous section 2.1.1, a Random Forest might comprise three decision trees, each utilizing a distinct pair of predictors, such as BMI and Diabetes, BMI and Smoker, and Smoker and Diabetes. This approach allows each tree in the Random Forest to capture different aspects or patterns within the data. Since individual trees only have access to a subset of variables and data instances, they are less prone to overfitting the training data. By averaging the predictions from multiple trees, a Random Forest effectively reduces the overall variance in the predictions. As the number of estimators (trees) in the ensemble increases, the model becomes more accurate and stable. When making predictions, each tree within the Random Forest provides its prediction for a given input instance. The final prediction is determined through a voting system, where the most frequently occurring output among all the trees becomes the collective prediction. Random Forests have gained widespread acceptance in the field of machine learning due to their robustness, ability to handle high-dimensional data, and their effectiveness in mitigating overfitting when compared to individual Decision Trees. They have been successfully applied in diverse applications, showcasing their versatility and utility across various domains.

2.1.2 Artificial Neural Networks

Artificial Neural Networks (ANNs) serve as a foundational machine learning technique that forms the backbone of what is commonly referred to as Deep Learning (DL) [66]. ANNs are engineered to mimic the functioning of the human brain. Essentially, they are depicted as graphs with specific mathematical attributes, where nodes correspond to neurons, and edges represent synaptic connections. The cornerstone of Neural Networks is the Perceptron [132], which functions as the fundamental algorithm governing information processing within the model. The core concept behind the Perceptron involves a linear combination of weights with input values, subsequently passing through an activation function:

$$f(x) = \sigma(\mathbf{w} \cdot \mathbf{x} + b) \tag{2.1}$$

Here, w denotes the model's associated weights, x represents the input values, and b signifies the model's bias. The function σ typically embodies a non-linear function, such as ReLU or sigmoid. A visual representation of this model can be found in Figure 2.3.

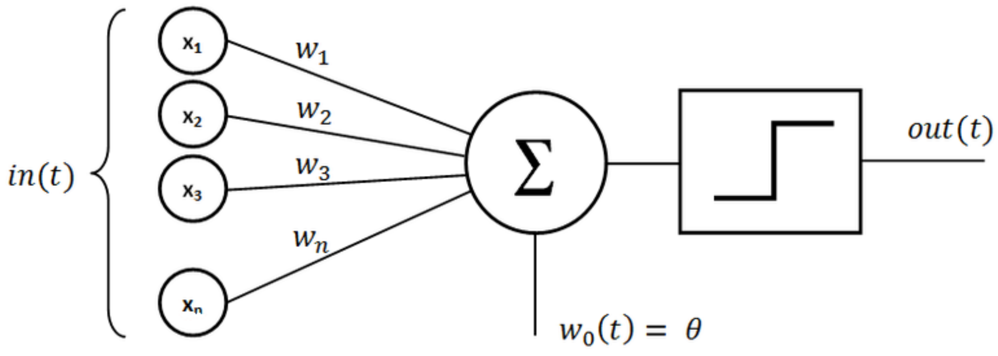


Figure 2.3: Perceptron

If we envision each neuron as represented by a perceptron, they can be organized to create diverse architectures based on the learning task at hand. Figure 2.4 illustrates a Feed-Forward Neural Network (FFNN), one of the most prevalent types of ANNs, where information flows from an input layer to the final neurons delivering the model's output.

In a broader context, the structure of a Deep Neural Network (DNN) can be summarized as follows:

- **Input Layer**, the input layer is the set of neurons processing the input data to the model.
- **Hidden Layers**, are the set of layers, which can vary in size, doing the computation and capturing complex patterns in the network.
- **Output Layers**, the set of neurons returning the output of the computation.

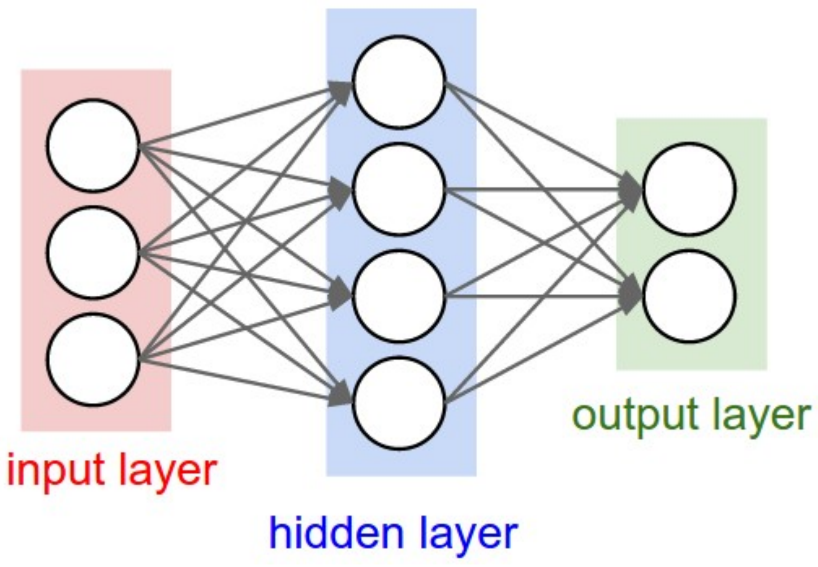


Figure 2.4: Deep Neural Network

The central algorithm underpinning the learning process for DNNs is Gradient Descent (GD) [66], an iterative optimization technique that determines optimal weights for the network based on an objective function, or loss function. Initially, the algorithm commences with an initial set of model parameters, encompassing weights and biases. Subsequently, it computes the gradient of the loss function for the parameters, revealing the direction of the steepest ascent in the loss. The model parameters are then adjusted in the opposite direction of the gradient, with the step size controlled by a learning rate. This process of computing gradients and updating parameters is repeated for a predefined number of iterations or until convergence or a defined stopping criterion is met. The process of updating the weights in the DNN is known as **backpropagation**.

More in general, DL models can be viewed as a family of parametric functions $g(\boldsymbol{x}|\boldsymbol{\omega})$, i.e. $\{f(\boldsymbol{x}|\boldsymbol{\omega})|\boldsymbol{\omega} \in \boldsymbol{\Omega}\}$ where the set $\boldsymbol{\Omega}$ is called *hypothesis space*. *Training* exploits data to find the parameter values in $\boldsymbol{\Omega}$ that minimize a so called *loss function*. In the general case, this is a function $\mathcal{L}(\boldsymbol{y}|\boldsymbol{\theta})$, where $\boldsymbol{y}$ corresponds to the output of DNN and $\boldsymbol{\theta}$ is the available empirical knowledge (e.g. the labels). For classical supervised learning, if we define the training set as a collection of n examples $D = \{(x_1, y_1), ..(x_n, y_n)\}$ where each instance is a pair (x_i, y_i) formed by an instance $x_i \in X$ and the corresponding label $y_i \in Y$, the goal of the learning task is to find a function $g : X \to Y$ mapping input to outputs. Training a leaner in this case means solving the following optimization problem, that is minimizing the loss function:

$$g^* = \arg\min_{g \in \boldsymbol{\Omega}} \sum_{i=1}^{N} l(g(x_i), y_i) \tag{2.2}$$

2.2 Knowldge Injection

One of the key strengths of DL models is their ability to automatically learn a *representation* of the features composing a data set. Broadly speaking, DNNs are *sub-symbolic* ML approaches that are very good at extracting useful information contained in large data sets. One of the advantages of DL techniques is that, in general, they do not rely on stringent assumptions on the distribution of the underlying data and on the function to be learned or approximated [1]. This allows them to be applied in many different areas to obtain high-quality results, without significant changes to the DNNs' structure and training algorithm.

However, there are contexts where purely data-driven models are not an ideal fit, e.g., in the presence of noisy and scarce data, which can make the learning process hard. In such situations, a great boost in performance can be obtained through the exploitation of domain knowledge or problem-specific information that can be used to improve the DL model simplifying the training process (for instance, structured data, knowledge about the data generation process, domain experts experience, etc). Hence, it makes sense to take advantage of domain information to improve the accuracy of DL models, so that they do not have to start from scratch while dealing with difficult learning tasks. In other words, *why learn again something that you already know?*

In general, the integration of domain knowledge in ML models is a multi-faceted topic that has been extensively explored from multiple angles. In-

[1]except for the assumption of independence and identical distribution of the samples

deed, already in the 1990s Towell et al.[153] introduced Knowledge-based artificial neural networks (KBANN) to integrate logic rules into neural architectures by mapping the rule's components to the neurons and weights. Similarly, different types of ML approaches have been considered as targets for injection of domain knowledge, for example, Support Vector Machines[99], probabilistic models[130], decision trees [151, 156], recommender systems[23], etc. The research area discussed in this section belongs to the wider paradigm of Artificial Intelligence aiming at merging sub-symbolic methods (such as DL models), and higher-level symbolic approaches (optimization, reasoning, first-order logic, etc)[37, 14, 36, 127].

In this department, we do not claim to provide a comprehensive and exhaustive overview of all the methodologies proposed in the literature (for a broader overview [159]), but we want to focus on a particular class of techniques for injecting prior information in DL models. Specifically, we consider prior knowledge that can be expressed in the form of *constraints* within the learning process of DNNs. This area has been relatively unexplored and this work aims at filling this gap.
In this context, we take into account constraints of different natures, ranging from First-Order and propositional logic predicates to linear and non-linear equations, where the constraints can represent relationships between the input features, relationships between input and output features, or bounds on the output variables. Moreover, we consider both hard constraints, setting conditions that must be satisfied, and soft constraints, setting conditions with an associated penalty if not satisfied.

2.2.1 Constraint-based Knowledge Injection in Deep Neural Networks

In this section, we highlight the main area of contribution and the specifics of the constraints encoding domain knowledge for DL.

Prior knowledge can be expressed as a set of logical constraints between the input features X and the target y. For instance, the monotonicity property holds if $x_1 \leq x_2 \implies y_1 \leq y_2$ for every pair in X. If we assume to have a learning task where the learner is trying to approximate a function g^* that maps an input X to an output y and a training set composed of examples (x_i, y_i), we define a set of constraints representing the monotonicity, π. Domain knowledge can be represented by algebraic equations, such as linear and non-linear equations, equality and inequality constraints, and logic formulas. These predicates can be applied to different groups of variables:

1. they can involve only the input features X, e.g. it is known that the

input features of well-formed data instances share particular properties, or are linked by a set of precise relations.

2. the constraints can concern only the output variables y, e.g. the output of the network must fall within a determined range.

3. the condition can involve both input x_i and output y_i, for instance, the real function to be approximated f could be monotonic ($x_1 \leq x_2 \rightarrow y_1 \leq y_2$).

As a particular case of constraints, we also include graphs (e.g. knowledge graphs), as they can be decomposed into collections of simpler constraints encapsulating the relations between nodes and edges.

2.2.2 Knowledge Injection Methods: A Taxonomy

The injection of prior knowledge within DNN learning processes can be classified into five categories, based on the injection mechanism (Figure 2.5):

1. *Constraints Learning*: constraints can be directly embedded in the DNN structure to govern its output – i.e., the target for the injection site is the function's family $\boldsymbol{\Omega}$ and the loss function $\mathcal{L}(\boldsymbol{y}|\boldsymbol{\theta})$; more often, the whole optimization problem to find $f(\boldsymbol{x}|\boldsymbol{\omega})$ is targeted by the approaches belonging to this class.

2. *Regularization Schemes*: constraints can be imposed, typically as "soft constraints", by adding them as regularization terms at training time (thus acting on $\mathcal{L}(\boldsymbol{y}|\boldsymbol{\theta})$); in these cases, the final loss function becomes a combination of multiple terms, that do not necessarily pull towards the same direction.

3. *Data Augmentation*: if the domain knowledge can be expressed as relational constraints among the features composing the training set X (both with or without labels), then it can be used to compute novel training examples, thus expanding the training data X^{aug}.

4. *Feature Space*: another class of techniques that operate on the training data X exploit prior knowledge to transform the original feature space to obtain an enriched one X' (e.g. feature engineering or creation of new features), where previously hidden information can be more easily extracted by the DL model; X' does not necessarily have a greater cardinality of X, whereas in the data augmentation class X^{aug} possesses a larger number of training examples compared to the original data set – nevertheless, these techniques are often used in conjunction.

5. *hypothesis space*: finally, domain-derived constraints can be used to prune the space of possible functions to be learned by considering only certain areas of the hypothesis space $\mathbf{\Omega}$, acting on the topology of the DNN and its hyperparameters.

These categories are not mutually exclusive, indeed, as we will see several methods are "hybrid approaches" exploiting multiple injection methods. Nevertheless, we decided to define a categorization of these approaches focused on the individual mechanisms that could be adopted for constraining the learning process.

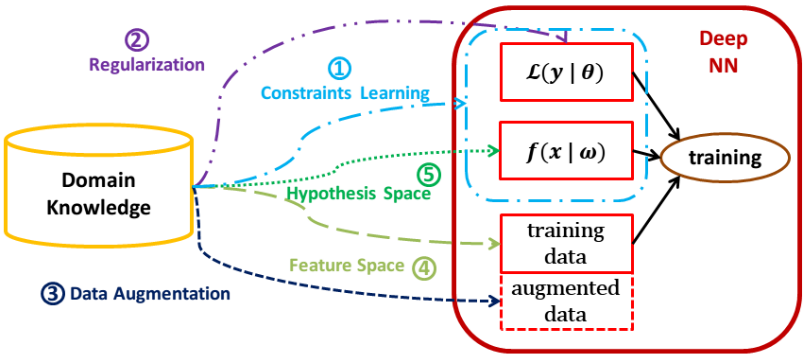


Figure 2.5: Domain Knowledge Injection in DNNs

2.2.3 Domain Knowledge Injection Approaches

In this section, we discuss recent methods for injecting prior information in DNNs, following the classification proposed in the previous section, we assign approaches from the literature to one of the following five classes: 1) constraints learning, 2) regularization schemes, 3) data augmentation, 4) feature space, 5) hypothesis space. Each subsection is devoted to one of the aforementioned branches. The classification of the methods analyzed in this survey is summarized in Table 2.2.

Learning with Constraints

Generally speaking, integrating constraints of various natures in a DNN means altering the structure of the neural network, often in conjunction with a modified training process. Besides the typical optimization problem of DNN training, enforcing additional constraints on the networks implies

Class	*Names*	*Methods*
Constraints learning	End-to-end Learning	Differentiable Layers
	Decision-focus Learning	Combinatorial Optimization
	Logic Constraints	Deep Declarative Networks
		Deep Reasoning Networks
		Logic Tensor Networks
		Deep Logic Models
		Relational Neural Machines
		Rule Knowledge Distillation
Regularization schemes	Semantic-Based Regularization	Regularization Terms
	Domain Adapted NN, LYRICS	FOL Clauses
	Adversarial Set Regularization	Hilbert-Schmidt criterion
		Propositional Logic
Data augmentation		Feature-side Information
Features space	Feature Engineering	Knowledge Graphs
	Feature Addition	Label Manipulation
	Feature Transformation	Samples Weighting
		Feature Relationships
Hypothesis space	Graph NN	Graph Embeddings
	Graph CNNs	Graph Convolutions
	LENSR	Logic Embedding
		Activation Functions
		Constrained Weights

Table 2.2: Classification of constraint-based domain knowledge injection methods

the resolution of a secondary optimization process, thus increasing the complexity of the overall DL model. In this context, Chen et al. [28] introduce *Deep Reasoning Networks (DRNets)*, which combine constraints reasoning and DNNs; DRNets encode the problem structure – expressed through a set of local and/or global constraints in the form of a MIP/CP optimization problem (e.g. as weighted constraints) – in the latent space of a encoder-(generative) decoder DNN. DRNet then minimizes a loss function composed of three terms: the regular reconstruction loss of an autoencoder, the satisfaction of local constraints (about single input data), and global constraints (about several input data points). As the constraint satisfaction problem can be (and often is) combinatorial and non-differentiable the authors proposed a set of continuous relaxations for different classes of constraints and an ad-hoc constraint-aware SGD algorithm. Alternatively, Vlastelica et al. [157] propose a DNN based on a black-box combinatorial solver, which optimizes a linear objective function. The core idea is to use the combinatorial algorithm as a building block of the neural network. This method is effective on classical problems with a linear cost function such as Traveling-Salesman, Shortest-Path, and Min-Cost-Perfect-Matching. The network learns a representation of the input instance, which is summarized by a vector w. The encoding, w, is then used by the solver to find the solution y, which minimizes the objective function, $c(w, y)$. However, since the solution set is finite, the loss function is represented by a piecewise constant mapping. This makes backpropagation harder because the gradient is either zero or infinite. The authors circumvent this issue with a continuous interpolation of the cost function, which produces a more informative gradient. They also introduce a λ hyper-parameter, which regulates the divergence between the interpolation and the original loss.

Stewart and Ermon [150] describe an approach to constrain a DL model coping with a supervised learning problem where very scarce labels are available. Their idea is not to rely on classical (x_i, y_i) samples, but rather to exploit domain information (e.g. physics laws) to specify constraints that should hold over the output space Y; in practice, the prior knowledge is modeled as a constraint function (they consider linear and non-linear equations and logical rules) $g : X \times Y \to \mathcal{R}$ that is then enforced on the output of the DNNs, penalizing structures not consistent with the prior information. To enforce this constraint, they use a different optimization problem w.r.t. to the one solved by Eq. 2.2, namely:

$$\widetilde{f}^* = \arg\min_{f \in \mathbf{\Omega}} \sum_{i=1}^{N} g(f(x_i), x_i) \tag{2.3}$$

In practice, they adopt an unsupervised approach and optimize for a necessary property of the output g, derived from domain knowledge. In addition, they also add a regularization term to Eq. 2.3 that can help the training process if some labels are available. The method is interesting and has shown promising results; however, finding the right function g and related regularization term is a very time-consuming task, and needs to be performed for each target domain.

Cotter et al. [33] propose a novel type of constraints, called *rate constraints*, which encapsulate prior knowledge on the domain (e.g. desired properties of the model output, such as fairness, coverage, recall, churn, etc.) by imposing bounds on the prediction made by an ML model. The concept of rate constraints was introduced by the same authors in previous works (see Goh et al. [65] and Cotter et al.[32, 34]) as constraints that can be expressed as a non-negative linear combination of positive classification rates and negative classification rates over different data sets. Imposing the rate constraints turns the training of a ML model into a constrained optimization problem. Goh et al. [65] considered only linear classifiers and linear constraints, which limits the expressive power of the approach. In most recent works, ([33]) they deal with non-linear classifiers and present a more efficient training algorithm. This algorithm is based on formulating the constrained optimization problem as a non-zero sum two-player game. If we use the same terminology employed in Eq. 2.2, the constrained optimization problem can be formulated as:

$$\begin{aligned} &\min_{\omega \in \Omega} g_0(\omega) \\ &s.t. g_i(\omega) \leq 0 \qquad for \quad i = 1, .., m \end{aligned} \tag{2.4}$$

where g_0 (the loss function) and g_i may be non-convex, and g_i may be non-differentiable as well; m is the number of constraints. A typical approach to solve Eq.2.4 consists of the method of Lagrangian multipliers, using the following Lagrangian:

$$\mathcal{L}(\omega, \lambda) g_0(\omega) + \sum_{i=1}^{m} \lambda_i g_i(\omega) \tag{2.5}$$

where λ_i is an $(m+1)$-dimensional vector of non-negative Lagrangian multipliers. Solving Eq.2.4 through the Lagrangian method can be framed as a two-player game where one player maximizes Eq.2.5 for the ML model parameters ω while the other maximizes it for the multipliers λ. The Lagrangian method is impractical for non-differentiable constraints, such as rate constraints; to overcome this limitation Cotter et al. propose a *proxy-Lagrangian* formulation where non-differentiable constraints are relaxed into differentiable versions *only when strictly necessary*, as opposed to other approaches from the literature ([62, 52]).

Gupta et al.[71, 54, 70] introduce *Deep Lattice Networks* (DLNs), models which learn monotonic functions using interpolated look-up tables. DLNs were offered as part of the TensorFlow python-based DL suite of tools [2]. DLNs are based on the concept of *lattice*, which is an interpolated look-up table that can approximate input-output relationships in the data; the look-up table overlaps a grid onto the input space and learns the values for the output in the vertices of the grid – the parameters of the DLN. The output values for the data points between the vertices are computed via interpolation. Lattices can be stacked on top of each other as layers, usually in conjunction with calibration layers that normalize the input space to fit in the acceptable range for the lattice. A very interesting propriety of DLN is interpretability: as the parameters of each lattice are the output corresponding to a particular input, they can be understood and interpreted more directly compared to typical DNNs. Lattices allow for the injection of *shape constraints* on the learned function, based on prior knowledge, such as I) monotonicity, II) convexity/concavity, III) unimodality (the learned function must have a unique peak or valley), and IV) pairwise trust (establish a semantic association between pairs of features). DLNs have been used in a variety of tasks, from imposing shape constraints to model learning set functions [31] and learning monotonic functions [173], to incorporate ethics principles into ML models[163].

In recent years, remarkable research efforts have been devoted to two relatively broad subcategories of knowledge injection methods that aim at directly enforcing constraints in DNNs: I) end-to-end learning and II) integrating logic formulas in DNNs. As there exist many research works fall into these two groups, they deserve separate subsections, as presented in the following.

End-to-end Learning

DL models are often employed in specific tasks, e.g. to find solutions to optimization problems. Usually, these approaches follow a two-step process where: first the model is trained with the observed data and then used to approximate an aspect of the optimization problem, e.g. the cost function. In these cases, the model's accuracy is not the only way to measure the performance of the approach, as task-related metrics might be more relevant.

Lately, a new paradigm emerged, namely *decision-focused learning* (or

[2]https://www.tensorflow.org/lattice/overview

end-to-end learning), where DNNs are trained to directly produce good results for the end goal of the whole task. In this context, prior information for the specific optimization task is introduced to guide the training of the DNNs used to approximate the needed functions. The most common type of optimization tasks are expressed as Mixed-Integer Linear Problem (MIP) or Constraint Programming (CP) models, as they arise in many diverse settings. In general, the DNNs for decision-focused learning produce a proxy (intermediate) solution for the target task, given a fixed parametrization of the network, then optimized by the loss function; however, as the solution space might be not continuous, differentiability issues might arise, which are in general solved through transformations, relaxations, and surrogates. For instance, Amos et al. [3] introduce *OptNet*, a way to embed exact constrained optimization problems in a DNN as a differentiable layer. The key contribution is that the output of each network layer is the solution of a constrained optimization problem based on previous layers (rather simply being a function, albeit complicated and non-linear) of the input, that is the previous layers. OptNet focuses exclusively on small quadratic optimization problems, as these problems can be efficiently used with GPUs. As DNN layers need to be differentiable, the derivative of the quadratic programming problems has to be computed. OptNet exploits I) implicit differentiation and matrix differential calculus to derive the gradients from the KKT matrix (Karush-Kuhn-Tucker conditions) of the quadratic problem and II) a novel GPU-based primal-dual interior point method to avoid the performance bottleneck in quadratic optimization layers (since current state-of-the-art quadratic programming solvers cannot solve multiple optimization problems in parallel on different mini-batches).

Another example of this class of methods was introduced by Wilder et al. [167], where it is applied to combinatorial optimization with the introduction of a continuous relaxation of the proxy solution (which is discrete), namely a *convex hull*; the loss is then computed by combining the gradient for the decision variables and the gradient of the model for its parameters. In another work of Wilder et al. [168] the focus is on graph optimization, especially on a link prediction problem; in this case, the differentiability is achieved with a specific architecture of the network (called *ClusterNet*), where through a *graph embedding*, performed using a Graph Convolutional Network, the graph representation is mapped into a continuous space and then used to produce the proxy solution. Similarly, Ferber et al. [56] proposed another approach stemming from the end-to-end learning field, namely they introduce *MIPaaL*, a method to encode MIP models in a NN. The key challenge to be overcome is the fact that MIP solution spaces are discontinuous and discrete, thus it is not trivial to differentiate through them. To cope with this issue MIPaaL uses a surrogate linear problem equivalent to the original problem; in this way, the MIP layer can be encoded as a network layer and backward propagation can be applied to the ensuing DL model. In particular, the authors of the paper use a cutting plane approach to iteratively solve the linear relaxation of the MIP problem, until an integer solution is found. A slightly different contribution is given by Donti et al. [50], which apply end-to-end learning on a stochastic optimization problem; however, in this approach, the focus is on the training process, where the gradient descent updates the network parameters using two functions: one computing the number of constraint violations and one represented by a classic loss function. If the proxy solution does not satisfy the constraints, the parameters are updated using the first function, otherwise using log-likelihood loss.

Gould et al. [67] introduce *Deep Declarative Networks* (DDNs), a class of end-to-end models whose network nodes are defined not by a forward processing function, but rather in terms of constraints and objective of mathematical programming (MP) problem. The optimization problem encodes the prior knowledge (e.g. underlying physical models) injected in the DNN. DDNs belong to the same mold of all recent approaches that aim at combining MIP/CP/MP optimization problems and DL models (e.g. [28, 56, 167]) and thus must face the same challenges; for instance, the requirement of differentiable layers for backpropagation in DNNs. Following the direction opened by Amos et al. [3], Gould et al. address the issue of casting the backpropagation as a bi-level optimization problem [41] and by employing implicit differentiation. Following a similar fashion, Pathak et al. [123] introduce an optimization procedure alternating between optimizing the DNN (via solving a convex optimization problem) and fitting a constrained distribution to the intermediate models; linear constraints on the output distribution are considered. Their approach is tailored for a specific task, namely weakly supervised segmentation in convolutional neural networks.

One more methodology for end-to-end learning is presented by Wang et al. in [162] with SATNet. This approach is based on the use of a differentiable MAXSAT (combinatorial equivalent of SAT problems) solver, which can be used as a layer in a more complex DNN. To this end, they propose an SDP (Semi-Definite Programming) relaxation of the MAXSAT problem, which can be solved optimally through block coordinates descent. The SATNet layer transforms the input variables, a binary assignment, into a continuous vector, which is then used in the SDP formulation. At last, the output layer converts the results produced by the solver into discrete assignments. This method has been employed to tackle challenging problems for deep classifiers, such as the parity function and Sudoku rules. Especially the latter gave promising results with high levels of accuracy; more importantly, the DNN solved this problem without any hand-coded knowledge of the problem structure, allowing for a sort of implicit logical reasoning.

Logic formulas as constraints

A particular class of constraints that can be used to force the behavior of a DL model consists of First-Order Logic (FOL) clauses. This area is ripe with potential research advances, as it aims at bridging the gap between the sub-symbolic (represented by the data-driven, black-box DL models) and symbolic layers (human-understandable knowledge expressed as high-level logic formulas). Logic Tensor Networks (LTN) were introduced by Serafini et al. [141] and are another approach to learning neural networks constrained by a set of First-Order Logic clauses expressing some prior relational information; LTNs express the FOL rules as a continuous relaxation of a logic rule using fuzzy logic. Li et al. [101] offers a different method to tackle problems where prior knowledge is expressed with First-Order logic formulas; in this case, the prior information is embedded directly in the DNN, where nodes, called *named neurons*, are labeled to mimic the logic elements in the formula; these are used to build *constrained neural layers*, which produce their output according to the truth value of each named neuron in the layer.

Marra et al. [108] propose a new framework to integrate FOL clauses in ML models, namely Deep Logic Models (DLMs), that are capable of jointly training a sensorial layer, in which the input pattern can be represented by videos, text, or images, and a reasoning layer, capable of enforcing structure to the model, i.e., integrate prior knowledge during the learning phase. DLM is composed of two phases or levels: the first one, the low-level, processes the sensorial input, while the high-level enforces the constraints representing the prior knowledge. The FOL formulas are converted into differentiable potential functions and integrated into the learning process. However, DLM does not specify how the low-level learner should integrate the supervised data. To this regard, Marra et al. [107] propose *Relational Neural Machines* (RNM), an approach that extends DLM, overcoming some of the ambiguities left out by the previous paper. In RNM a probability distribution is established over the set of target variables, the output of one or more sub-symbolic learners. RNM is composed of two phases, similar to DLM. The high-level output variables are defined by a conditional probability function (exponential family), where the outputs are conditioned by both the low-level patterns identified by the sub-symbolic learners and by the constraints encapsulating high-level relationships between input and outputs (expressed through FOL formulas).

Another approach integrating logic formulas in DL models has been proposed by Hu et al.[78], following previous works by Hinton et al.[75]. The core of the approach is a *rule knowledge distillation* mechanism that exploits the information contained in a set of FOL rules to train a deep NN. This is done by forcing the NN to mimic the predictions of a rule-regularized teacher; both the "student" network and the teacher evolve during the training. The teacher is built at each training iteration by projecting the current predictions of the student into a latent space constrained by FOL rules (thus having desired, knowledge-inspired proprieties); the teacher is trained to respect the constraints imposed by logic rules and to produce predictions close to the students' ones – this is a convex optimization problem solvable in closed-form. Then, the student loss function is composed of two terms, one corresponding to the imitation of the soft labels provided by the teacher and one trying to predict the true labels (with standard loss function such as cross-entropy for classification).

Fischer et al. [60] introduced an approach similar to the end-to-end learning paradigm, but more general, called *DL2*. DL2 is a framework for explicitly embedding constraints in DNNs, specifically, it allows to translation of logical formulas into loss functions, i.e. by defining recursively the corresponding mathematical equation for each term in the formula; the training is then carried out using projected gradient descent. The model is also provided with a SQL-like query language, which allows the interrogation of the network on a specific input, to: check if the constraints are satisfied and train the model for input outside of the set of observed data; basically, the network can be challenged with queries that help improve its accuracy beyond the data available for the training – this method is called *global training*.

Regularization Schemes

Another common way in which prior knowledge can be incorporated in DNNs is by specifying an apriori preference for certain functions f among all the ones allowed by the hypothesis space ω). For instance, the regularization term can express a preference for "simpler" models; regularization techniques are widely used to prevent overfitting (e.g. Dropout [149]) when training deep networks with a very large number of parameters. Regularization can also be exploited to inject domain knowledge. To do so, some form of prior insight, i.e. constraints, is translated into a regularization term able to measure the level of consistency with the domain knowledge and guide the loss optimization process. More in general, the loss function (in the case of supervised learning tasks) can be defined as follows (extending the definition of Eq. 2.2):

$$f^* = \arg\min_{f \in \mathbf{\Omega}} \sum_{i=1}^{N} \lambda_\tau L(f(x_i), y_i) + \lambda_\pi L_\pi(f(x_i)) \tag{2.6}$$

where L represents the true loss, e.g. Mean Squared Error (MSE), while L_π represents the regularization term given a set of constraints π; these terms are weighted, respectively, with λ_τ and λ_π; the choice of these weight parameters is non-trivial and might consistently affect the outcome. Fioretto et al. [59] address this problem by exploiting *Lagrangian duality*; the idea is to optimize the regularization multipliers during the training phase of the model. In particular, after each training epoch, the λ value is updated using the dual ascent rule. This provides an automated and sound tool to tune the weight parameters in regularization schemes.

This class of approaches shares some aspects with end-to-end learning, as the penalty terms in the loss function can be applied also to end-to-end learning. However, since in this case, the DNNs are not necessarily used within larger optimization models, the mechanism to influence their behavior is different. In the last few years, many authors worked on this class of methods propose a variety of approaches to the problem; for instance, Muralidhar et al [117] developed a DNN, called *domain adapted neural network* (DANN), which exploits a mathematical formulation of the constraints, namely approximation and monotonicity constraints, whose degree of satisfaction is used as a regularization term. Another method, proposed by Demesteer et al. [40], provides a means to inject propositional implication constraints into convex loss functions. The assumption is that every relation refers to a set of tuples, which represent the training instances constrained by the implication. The logic proposition can then be translated into a summation which defines if the enforced prior knowledge is satisfied. To make the process more efficient the tuples are to a non-negative embedding space, more specifically are bounded to a hyper-cube in which they can assume values in $[0, 1]$. A more general approach is introduced by Diligenti et al. in a couple of research works [48, 47] and it was called *Semantic Based Regularization* (SBR), which provides a set of rules to translate First-Order logic formulas into fuzzy constraints, then used as penalty factors in the loss function. This is achieved by introducing a *t-norm* function, which can be defined in different ways, according to the desired interpretation of the domain constraints involved in the regularization, allowing for an adaptable tool. Marra et al. [109] continue along this line of research and present a strongly related method denoted *LYRICS*, a framework that introduces a declarative language to express the domain knowledge and enforces it using SBR on top of DNNs, allowing a very flexible representation of the constraints and the prior information.

A stochastic approach with semantic loss is proposed by Xu et al. [171], which uses a regularization term given by the probability of generating a state satisfying the domain-based constraints; during the training process, the presence of states not satisfying the desired constraints is penalized acting on the loss function. Remaining in the context of semantic regularizers, Silvestri et al. [146] describe an SBR-inspired technique for injecting domain constraints in a DNN used to extend a partial variable assignment for the Partial Latin Square problem, specifically finding feasible solutions that respect domain constraints.

Minervini et al., in [113], present an approach based on adversarial networks, namely *Adversarial Set Regularization* (ASR). The method defines the injective knowledge through FOL clauses, which are used to derive an inconsistency loss; this function measures the degree of violation of the constraints previously defined. The model is composed of two networks: the adversary is trained to produce a set of instances which maximizes the inconsistency loss, while another model, called discriminator, uses the adversarial input to regularize the training process, therefore enforcing the initial clauses. The authors propose this model underlining how the adversarial process can be seen as an adaptive regularizer.

Takeishi et al. [152] devised a generative model that exploits the prior knowledge deriving from feature dependence between variables. Assuming that some features are going to be more dependent on one another, the paper introduces a way to evaluate the degree of statistical independence (or dependence) between variables; for this purpose, the authors use a kernel-based measure, namely the Hilbert-Schmidt criterion (HSCI)[69]. Feature with high statistical dependence will be characterized by smaller HSCI values, hence the difference between highly dependent and highly independent couples of variables should always be a negative value. The regularization terms are designed to exploit this property: the penalty factor for the loss function is the summation of all the instances in which the feature dependence constraint is violated.
In particular, given a set of tuples:

$$K = \{(x_i, x_i^+, x_i^-) | i = 1, .., n\}$$

in which x_i^+ is more statistically dependent on x_i than x_i^- is; the loss function can be written as follows:

$$L(f(x_i), y_i) + \frac{1}{n}\sum_{i=1}^{n} max(0, HSCI(x_i, x_i^+) - HSCI(x_i, x_i^-))$$

Data Augmentation

Besides working on the feature and the hypothesis space, another mechanism to infuse the domain knowledge in DNNs regards the training data, mainly in the forms of creating *ex-novo* entire training sets or adding new examples to existing ones following criteria defined by the domain knowledge (for instance, examples respecting certain relationships among the input features). We refer to these methodologies with the term *data augmentation*. Data augmentation approaches based on prior information (constraints) have been recently explored, especially to cope with data sets of limited size and the related issue of poor generalization performance [73, 74].

Data augmentation techniques have a strong history of success in the context of image-based learning tasks (e.g. image classification) [160]. For image classification tasks, the training set can be augmented by applying a plethora of transformations to the images in the original training set, thus feeding the NN to be trained with a more varied set of examples. The selection of the best transformations to apply to augment the available data is a process that is typically guided by information obtained from domain experts. In particular, constraint-based domain information can be used to augment the available data, e.g. linear and non-linear functions such as rotation, distortion, flipping, etc.

For instance, Bjerrum et al. [16] propose a data augmentation technique to train a Long-Short Term Memory (LSTM) model for classifying chemical molecules. Each molecule can be described as a combination of its composing elements, encoded as a concatenation of strings. A single molecule has multiple possible encoding strings; the authors propose to expand the training set by enumerating the chemically allowed combinations for each data point/molecule. The enumeration takes place via a heuristic algorithm that enforces on the generated examples the same chemical properties of the original data points; these properties are encoded as a collection of constraints (linear combinations representing admissible chemical properties) among the input features (the concatenated strings). In this case, the constraints involve both input features x_i and the output y_i, as the relationships among X are used to generate novel data points with the same output value (the label).

Mollaysa et al. [114] describe a different methodology for data set augmentation: they consider *feature side-information*, domain knowledge describing feature properties, and/or feature relations. The feature side-information is expressed as a matrix whose rows represent the prior information associated with each feature; a similarity function is introduced to compute the pairwise similarity of different features. An augmented training data point is obtained from an original example by applying a transformation that preserves the associated label and perturbs the values of similar features. Again, the focus is on constraints among input features x_i.

Similarly, Vo et al. [158] devise a data augmentation method based on similarity scores among features to improve the results of CNNs used in sentiment analysis. In this case, the original data set is composed of labeled sentences (the training examples); each sentence can be decomposed into a set of sentiment terms, i.e. the features. The authors introduce a similarity measure for the sentiments and then propose an algorithm (based on quadratic programming) to generate similar sentences from the original ones, based on the sentiment similarity score; the augmented similar examples are annotated with the same label as the corresponding original training points.

Feature Space

The performance of DL models strongly depends on the quality of the available training data, either labeled or not; the features in the data define the *feature space* whose implicit information is extracted by DNNs. The shape of the feature space is a critical issue for the performance of DL models and the ease of their training. For instance, *feature engineering* is a common method for improving the accuracy of purely data-driven ML models by selecting useful features and/or transforming the original ones to facilitate the learner's task. In general, this is a difficult problem and requires much effort, both from system experts and ML practitioners [93]. In recent years, several research avenues studied the possibility of automatically exploring the feature space to extract only the most relevant features, with most of the approaches belonging to the AutoML area [118] and reinforcement learning [94].

The majority of current feature engineering methods aim at selecting the optimal features for a specific learning task and generally tend to reduce the feature space, by selecting only the most relevant features (feature extraction or feature compression). Furthermore, these methods are generally purely data-driven and require large amounts of data; although they aim at exploring the feature space in an efficient way when the number of features is large this becomes a nontrivial problem.

A relatively unexplored direction is the use of domain knowledge to create novel features that render *explicit* the information hidden in the raw data. This is a form of feature space extension to highlight the prior knowledge embedded in the original features but not easily extractable by a neural network. In practice, the approaches proposed in this area work on the original input features X to generate an extended feature set $X' = X \cup \{x_j\}$, where $\{x_j\}, j \in 1, ..N_j$ is the set composed by the additional features (N_j is the number of added features). The new features x_j are computed as a combination (linear or non-linear equations) of the original ones, depending on the domain constraints.

For example, Atzmuller et al. [4] enrich the feature space using the domain-specific information encoded by knowledge graphs. The relationships explicitly described by the knowledge graph represent constraints (soft and hard) among the original input features and can be used to create additional features, which then improve the accuracy of a supervised DNN. Similarly, Miao et al. [112] leverage domain-based features to increase the feature space and boost the performance of a DL model. They do not use the knowledge graph to obtain the additional features but rather an ensemble of decision trees solving a classification task on a subset of the data devoted to the training of the DL model. Each tree learns domain-specific information and partakes in the final prediction through its score; these scores are then added as additional features to the training set.

Kamiran et al. [88, 90, 89] enforce constraint satisfaction in the training data via a pre-processing step. They work with relational constraints and manipulate the feature space by changing the labels or the weights associated with the training examples. The approach enables the use of standard DL methods with no modification (no costly exploration of the hypothesis space) and can deal with constraints on large sets of examples. The main limitation is the risk that bias in the model or the training algorithm may prevent getting close to the pre-processed labels.

Another method to incorporate domain knowledge by enriching the feature space is discussed by Berrar et al. [13], which study the improvement of a DNN for the prediction of soccer match outcomes obtained through the addition of domain-inspired features. The training set is a time series composed of matches among two different teams and related outcomes; their approach consists of adding a set of novel features that encapsulate i) the rating of the teams involved in the match and ii) the results obtained in the last k matches by each team. These novel features are encoded as a set of linear and non-linear equations applied to the original input X. Borghesi et al. [19] shows how expanding the feature set with additional features discovered thanks to domain knowledge can improve the prediction accuracy of a DL-supervised regression model used in the context of transprecision computing. The key idea is that the input features are connected by binary constraints, namely inequalities and that by rendering these connections explicit with the addition of new features, the learning task is simplified.

Hypothesis Space

A DNN can be characterized by its position in the so-called *hypothesis space*, namely the multi-dimensional space covering its structure and its hyperparameters. The architecture of a NN is a very relevant factor for determining its performance on different learning tasks. The hypothesis space has been explored with implicit guidance provided by domain knowledge for many years, as attested by the introduction of convolutional networks, whose structure is based on the locality assumption (e.g. pixels close to each other in an image are related). Implicit knowledge about temporal locality has also led to a wide range of architectures targeted at handling time series and sequences, for instance, recurrent NNs (RNNs), Long-Short Term Memory NNs (LSTMs), and Temporal Convolutional Networks [7].

In more recent years, a remarkable research effort has been devoted to exploring DNN architectures optimized for circumstances where the domain knowledge can be expressed in the form of graphs, precisely with a DL model called *Graph Neural Network* (GNNs), proposed by Scarselli et al.[139]. *Graph Convolutional Networks* (GCNN) were introduced in two different variants by two research groups: Kipf et al. [96] and Defferrard et al. [39] Similarly to GNNs, the purpose of GCNNs is to exploit the same type of graph-expressible prior information. GNNs have been used in several fields [175], owing to their capability to deal with data whose structure can be described via graphs, thanks to a generalization in the spectral domain of the convolutional layers found in many deep learning networks. The most common applications of GCNNs involve semi-supervised classification tasks to predict the class of unlabeled nodes in a graph – a case of graph learning. However, graph convolutions have been applied to disparate learning tasks from several areas, from image classification[102] to physics predictions [115].

Using prior information expressible as a graph has been proposed also to devise other types of NNs, namely networks whose structure resembles the LSTM's but where the nodes connections are determined by the prior information [106]. Similarly, Jain et al. [83] combine the temporal structure of RNNs with spatial-based information, namely, the domain information is encoded as sequences of actions each one characterized by a time and a position in space, which are then represented through an extended RNN. As mentioned earlier, domain knowledge represented as a graph can be expressed by sets of constraints, which encode the relationships between the nodes through the edges.

Another potential target for imposing constraints in the hypothesis space is the weights of the DNN itself. For instance, Ayinde et al.[5, 6] constrain the weights of a sparse autoencoder network to be greater or equal to zero (non-negative); this in turn, improves the sparsity and reconstruction quality in the autoencoder (as already demonstrated by Hosseini et al.[77]. Similarly, constraints can be imposed on the activation functions of DNNs. Exploiting this idea, Frerix et al. [61] develop a framework for incorporating homogeneous linear inequality constraints on neural network activations. The authors impose homogeneous linear inequality constraints by splitting this task into two phases, an initial feasibility step (when a suitable parametrization of the constraint set is pre-computed) and an optimization step during training. This enables training neural networks that are guaranteed to satisfy non-trivial constraints on the neurons in a scalable manner and demonstrate this experimentally on a generative modeling task.

As noted in other points of this review, the distinction between different injection method types is not always clear-cut. For instance, Bansa et al.[11] propose an approach that addresses the hypothesis space via a regularization strategy (previously mentioned in 2.2.3). They consider CNNs and, more, in particular, the struggle associated with their training; in their paper, they discuss the imposition of *orthogonality* constraints on the linear transformations between the hidden layers of a CNN. The idea is that orthogonality implies energy preservation among different layers and this preservation stabilizes the activation over the CNN layers, thus improving the network accuracy and reducing convergence time. Similarly, Yaqi et al. [172] describe a knowledge injection method that cannot be precisely pinned down into a specific category. Specifically, the authors discuss how to embed symbolic knowledge expressed as propositional logic formulas using both an ad-hoc GCNN and a semantic-based regularization term added to the loss function; the proposed method is called Logic Embedding Network with Semantic Regularization (LENSR). Yagi et al. treat the domain knowledge expressed as a collection of logic formulas viewing them as graph structure, generating "logic graphs" that they then embed in the DNNs using graphical convolutions. In addition to this graph embedding, LENSR employs a semantic regularization term added to the loss function, which encourages the logic formulae embeddings to be far from unsatisfying assignments and close to satisfying ones. LENSR is validated on both a synthetic data set and on a real-world task, Visual Relation Prediction[3], showing a neat performance boost.

2.2.4 Physics-Informed Machine Learning

Physics-informed Machine Learning has gained high attention in the last few years [29, 98, 22, 128, 174, 126, 120], enabling the integration of physics knowledge into machine learning models. The key idea is to incorporate known physical laws and principles into the structure and training process of the ML model. In this section, we briefly present some of the most promising trends in Physics-informed Machine Learning. For an exhaustive literature overview, we refer the reader to [91].

[3]The goal of VRP is to predict the correct relation between two objects given visual information in an input image

Physics-informed loss function.

The most straightforward way to enforce constraints in Neural Networks is via an additional term in the loss function. In [92] the authors propose a Physics-guided Neural Network, a framework that exploits physics-based loss functions to increase deep learning models' performance and ensure the physical consistency of their predictions. Similarly, the work of Chen et al. [85] generalizes Recurrent Neural Networks adding a regularization loss term that captures the variation of energy balance over time in the context of lake temperature simulation. Work of [15] proposes to enforce physics constraints in Neural Networks by introducing a penalization term in the loss function defined as the mean squared residuals of the constraints.

Physics-informed neural architectures.

Recent works focus on designing deep learning frameworks that integrate physics knowledge into the architecture of deep learning models [29, 98, 22, 128, 174, 126, 120]. Neural Ordinary Differential Equations (Neural ODEs) [29] bridge neural networks with differential equations by defining an end-to-end trainable framework. In a Neural ODE, the derivative of the hidden state is parameterized by a neural network, and the resulting differential equation is numerically solved through an ODE solver, treated as a black-box. Neural ODEs have proven their capacity in time-series modeling, supervised learning, and density estimation. Moreover, recent works adopt Neural ODEs for system identification by learning the discrepancy between the prior knowledge of the physical system and the actual dynamics [98] or by relying on a two-stage approach to identify unknown parameters of differential equations [22]. Recently, O'Leary et al. [120] propose a framework that learns hidden physics and dynamical models of stochastic systems. Their approach is based on Neural ODEs, moment-matching, and mini-batch gradient descent to approximate the unknown hidden physics. Another approach is represented by the Physics-informed Neural Network (PINN) framework [128] which approximates the hidden state of a physical system through a neural network. The authors show how to use PINNs both to solve a PDE given the model parameters and to discover the model parameters from data. Zhang et al. [174] further extend PINNs by accounting for the uncertainty quantification of the solution. In particular, the authors focus on the *parametric* and *approximation uncertainty*. The first is accounted for by considering the parameters of the differential equations as represented by a stochastic process and using the neural components to learn the arbitrary polynomial chaos expansion of its solution from data. The approximation uncertainty is quantified by relying on *dropout*. Universal Differential Equations (UDEs) [126] represent a generalization of Neural ODE where part of a differential equation is described by a universal approximator, such as a neural network. The formulation is general enough to allow the modeling of time-delayed, stochastic, and partial differential equations. Compared to PINNs, this formalism is more suitable to integrate recent and advanced numerical solvers, providing the basis for a library that supports a wide range of scientific applications. UDEs also represent a generalization of Neural ODE where part of a differential equation is described by a universal approximator, such as a neural network. The formulation is general enough to allow the modeling of time-delayed, stochastic, and partial differential equations. Compared to PINNs, this formalism is more suitable to integrate recent and advanced numerical solvers, providing the basis for a library that supports a wide range of scientific applications.

$$\mathcal{N}\left[u(t), u(\alpha(t)), \mathrm{W}(t), U_\theta(u, \beta(t))\right] = 0 \tag{2.7}$$

where u is the system state, U_θ is a universal approximator parametrized by θ, $\alpha(t)$ is a delay function and $\mathrm{W}(t)$ is the Wiener process.

System identification.

Research towards the automated dynamical system discovery from data is not new [35]. The seminal works on system identification through genetic algorithms [18, 140] introduce symbolic regression as a method to discover nonlinear differential equations. However, symbolic regression is limited in its scalability. Brunton and Lipson [26] propose a sparse regression-based method for identifying ordinary differential equations, while Rudy et al. [135] and Schaeffer [72] apply sparse regression to PDEs discovering. Recent works [105, 98, 22] focus on applying physics-informed neural architectures to tackle the system discovery problem. Lu et al. [105] propose a physics-informed variational autoencoder to learn unknown parameters of dynamical systems governed by partial differential equations. The work of Lai et al. [98] relies on Neural ODE for structural-system identification by learning the discrepancy for the true dynamics, while Bradley at al. [22] propose a two-stage approach to identify unknown parameters of differential equations employing Neural ODE. The work of [105] proposes an encoder-decoder architecture to learn the parameters of a differential equation from data. The dynamics encoder (DE) is a deep convolutional neural network that extracts the latent variables associated with the parameters of the differential equations. The propagating decoder is a deep convolutional neural network with a residual connection that predicts the next state given the current one and it acts as an integral solver.

System identification refers to a set of methodologies aimed at designing mathematical models of dynamical systems from measurements.

Work of [26] developed Sparse Identification of Nonlinear Dynamics (SINDy), a data-driven method to learn nonlinear terms of differential equations. Given measurements about the system state $\boldsymbol{X}$ and its derivative $\dot{\boldsymbol{X}}$ and a set of candidate basis functions $\boldsymbol{\Theta}(\boldsymbol{X})$, the problem is framed as the following regression task:

$$\dot{\boldsymbol{X}} = \boldsymbol{\Theta}(\boldsymbol{X})\Xi \tag{2.8}$$

where Ξ is the set of coefficients associated with the basis functions.

Several works improve or extend the SINDy method [25], [87], [134], [21], [38], [170], [30]. For example, in [134] the authors propose SGTR, an improvement of SINDy to discover time-dependent parameters of PDE.

Neural ODEs have been recently adopted for system identification. [98] relies on Neural ODE for structural-system identification, employing a Neural Network to learn the discrepancy w.r.t. the true dynamics. Similarly, [22] proposes a two-stage approach to identify unknown parameters of differential equations employing Neural ODE.

Work of [105] proposes an encoder-decoder architecture to learn unknown parameters of a differential equation from data. The dynamics encoder (DE) is a deep convolutional neural network that extracts the latent variables associated with the parameters of the differential equations. The propagating decoder is a deep convolutional neural network with a residual connection that predicts the next state given the current one and it acts as an integral solver.

2.3 Epidemics

Epidemics can be described as a physical phenomenon in which a pathogen, whether it's a virus, bacteria, or another infectious agent, rapidly spreads within a population. This phenomenon reflects a dynamic interplay between the pathogen's contagiousness and the vulnerability of the host population. As the pathogen proliferates, it can lead to a sudden surge in the number of individuals affected by the disease, often surpassing typical infection rates. Epidemics highlight the urgency and complexity of infectious disease outbreaks, necessitating rapid responses from public health authorities and the implementation of measures to control the pathogen's expansion, protect public health, and minimize societal impact. Humans have attempted to comprehend and model these intricate phenomena for thousands of years. Eventually, the field of epidemiology, the study of epidemics, emerged in the

19th century, notably with Dr. John Snow's pioneering efforts to map the spread of cholera in London.

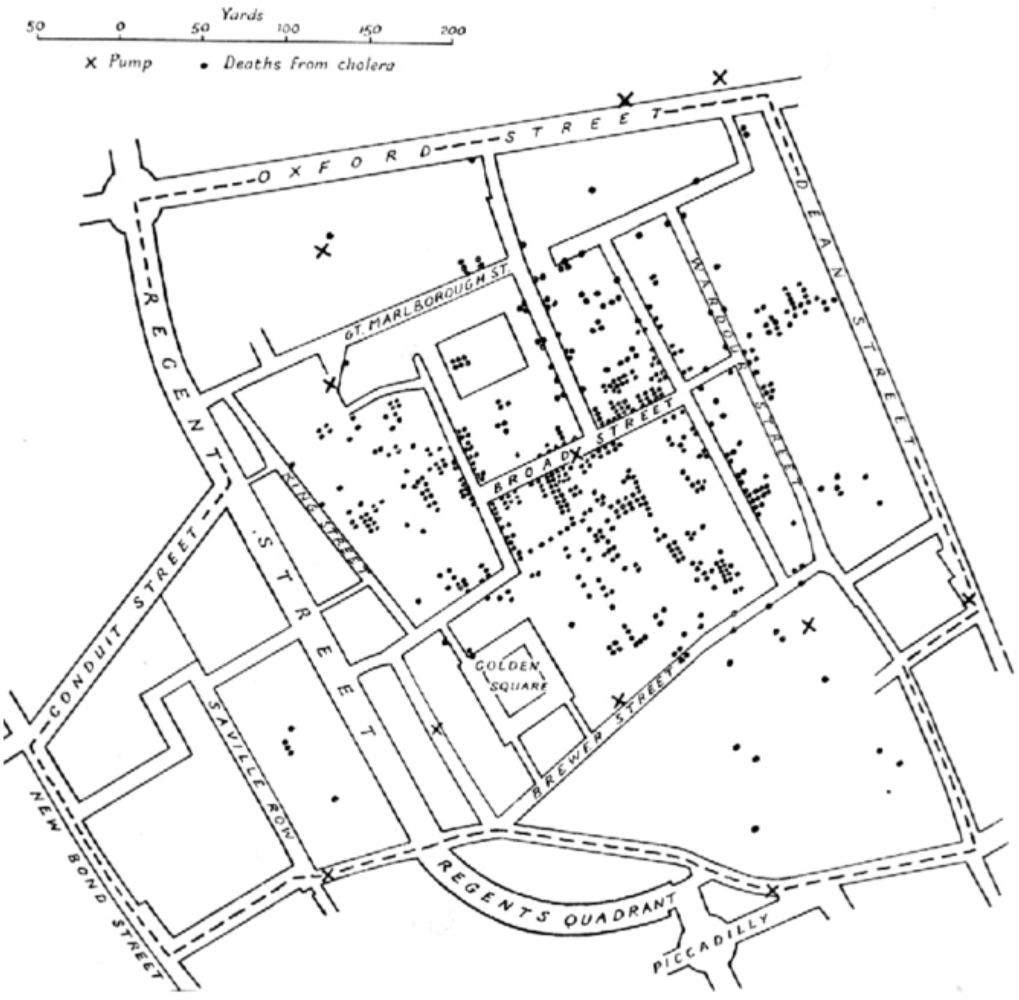


Figure 2.6: Snow cholera map showing cases of cholera in the London epidemics of 1854, clustered around the locations of water pumps.

2.3.1 Epidemiological Models

Epidemiological models are simplified representations of disease outbreaks, aiming to depict the dynamics of the disease. In the past century, there has been a proliferation of diverse frameworks that have adopted various approaches to address this issue. In this context, it is important to acknowledge that epidemics are complex systems, characterized by numerous interconnected and interdependent components that manifest intricate behaviors and properties due to these interactions. Complex systems include ecosystems, economics, and climate, making modeling these phenomena exceptionally challenging. Next, we will outline some of the approaches that have been adopted over the years to address this issue:

Compartmental Models

Compartmental models provide a mathematical framework for understanding the dynamics of disease outbreaks. The fundamental concept behind this approach involves dividing the population into distinct compartments or categories that describe an individual's state concerning an ongoing epidemic. For instance, an individual can fall into one of three categories: susceptible (never infected but lacking immunity), infected (actively carrying the illness and potentially spreading it), recovered (having overcome the disease), or deceased. This simplification is commonly referred to as the SIR model (Susceptible-Infectious-Recovered). The transitions between these categories are described by a set of parameterized differential equations:

$$\frac{dS}{dt} = -\frac{\beta \cdot S \cdot I}{N} \tag{2.9}$$

$$\frac{dI}{dt} = \frac{\beta \cdot S \cdot I}{N} - \gamma \cdot I \tag{2.10}$$

$$\frac{dR}{dt} = \gamma \cdot I \tag{2.11}$$

Here, β represents the infection rate, calculated as the average number of contacts per person during each time step multiplied by the probability of disease transmission in contact between a susceptible and an infectious individual. Similarly, γ denotes the recovery rate, defined as the reciprocal of the recovery period, and N signifies the total population. Complex variations of this basic model can be developed by introducing additional categories, such as the SEIR model (Susceptible-Exposed-Infectious-Recovered) or the SIRD model (Susceptible-Infectious-Recovered-Deceased). More recently, the SIDARTHE model has been proposed to better represent the dynamics of the COVID-19 virus by extending the number of compartments and the interactions among them [63].

Renewal Equations

Renewal Equations [27, 53, 104], are utilized for forecasting the count of newly infected individuals within a susceptible population. The foundational concept of this methodology is that each infected person is responsible for propagating a new infection following a certain duration, termed as the generation time. Expressed in a general form, the renewal equation model is denoted as:

$$y_{t+1} = R_t \int_0^t y_{t-s}\ \omega(s)ds \tag{2.12}$$

Here, y_{t+1} signifies the number of individuals infected at the instance $t+1$, R_t denotes the rate of infection at time t and $\omega(s)$ characterizes the distribution of generation time. The integration spanning from 0 to t captures the cumulative impact of all preceding infections on the current count of infected individuals.

The generation time distribution provides insight into the temporal interval between successive infections caused by an individual carrier. This distribution is pivotal for understanding the dynamics of the infection's spread and assists in making more accurate predictions and forming effective control strategies.

2.3.2 Deep Learning for Epidemics

In this survey, *we focus on the application of DL models to approximate the dynamics of diseases and to produce forecasting systems capable of predicting the spreading of an airborne virus.* We decided to keep the scope of the considered illnesses as wide as possible. However, it is impossible not to notice the wealth of many recent works regarding COVID-19. Nevertheless, the approaches outlined in this section can be applied to any disease if proper modifications are made. One of our goals is to provide a general taxonomy of the different DL methodologies that can be deployed to tackle this problem. The related scientific literature follows two main directions: one is focused on the application of *pure* DL models, in either a *vanilla* fashion or with a combination of different Artificial Neural Networks (ANN), in an attempt to make predictions with minimal assumptions or human intervention; the other one tries to incorporate well-established epidemiological methods with DL, to take advantage of decades worth of models and insights, and produce more explainable predictions. In particular, the second class of methods relies on the idea of embedding domain-specific knowledge into the predictive model, a new trend in the field of ML. The principle at the foundation of this approach is to integrate into the learning process prior information, or techniques, relative to the field of interest to improve performances and build custom models (e.g., *end-to-end learning*).

Vanilla Deep Learning

In this section, we revise the application of traditional DL methodologies to predictive epidemiology. Here we observe the prevalence of two perspectives: on one side, the application of techniques historically designed to model sequential data, therefore particularly suited in the presence of temporal data (i.e., time series), while on the other, the implementation of DL approaches for spatial information transformed to handle temporal information.

Recursive Neural Networks. Traditionally, this class of methods has been widely applied to approximate the dynamics of epidemics (and more in general sequential data), thanks to their structure and capability of developing a *long-term memory.* In the RNNs family, we include Long-Short Term Memory Networks (LSTM) and Gated Recursive Units (GRU), which are particularly suited for temporal series. The standard application of RNNs involves the construction of an appropriate dataset by arranging each feature according to the number of *look-back days*, namely, how many days in the past are taken as input when making a prediction. This allows incorporation in the forecasting process of information regarding the general trend of the epidemiological curve.

A recent application of RNNs can be seen in [142], where authors have used LSTM and GRU to model the COVID-19 pandemic data (i.e., number of infected, deceased, and recovered cases) of different countries, with results that show how these methods outperform more classical ML approaches in terms of prediction accuracy on historical data, such as Support Vector Regression (SVR), Radial Basis Function (RBF) kernels and Auto-Regressive Integrated Moving Average (ARIMA). In the same study, the authors improved the model using Bidirectional LSTM (Bi-LSTM) to enhance the capability of developing long-term memory in the model. While classical RNNs-based approaches process sequential data in a specific order, i.e., from past to future, Bi-LSTM allows the information to flow in both directions of the time series (i.e., past-future, future-past), with an increase in performance, also w.r.t. LSTM and GRU.

In [143] is proposed a method to bring further the application of LSTM on epidemiological data. The idea is to stack multiple LSTM layers, where each intermediate layer output is used as input for the next one, resulting in more complex and deeper models. This method, called Stacked-LSTM, seems to increase significantly performance due to its higher capability to abstract the input through the stratification of the layers.

While Recursive networks produce an accurate approximation of time series, they often do not include contextual information, such as spatial or seasonal data, that might be relevant to consistently capture the dynamic of a virus. This issue is directly tackled in [155], where authors proposed an LSTM-based method incorporating geographical proximity information and climate variables (e.g., humidity, temperature, precipitation) to predict the number of flu cases in the population. The approach consists of two steps: in the first stage, the LSTM neural network is trained on the flu time series, while in the second stage, the impact of climatic variables and spatio-temporal adjustment is added to the flu counts estimated by the LSTM model. The *contextual features* are integrated into the learning process via a linear combination of the variables and multiplied with the actual predicted number of daily new cases.

Convolutional Neural Networks. This DL tool is one of the most widely used approaches to deal with spatial proximity within input data. Such aspects can be very useful in forecasting epidemiological trends. As CNNs have been proved useful at dealing with spatial proximity (by extracting features embedding the spatial locality), they can be adopted for dealing with temporal locality as well, as in the case of ordered sequential data.

Most basic CNN applications rely on the use of ordered time series as input and require the model to predict the next data points in the series; however, CNNs are known to be prone to overfitting historical data. Nevertheless, this can be sufficient in the presence of poor contextual data to outperform LSTM or simpler algorithms such as decision trees (DT) [97]. Predictions can be improved by feeding the models with multiple correlated time series. For example, in [79] authors use the data relative to confirmed, deceased, and recovered cases to predict the daily count of infected. The CNN is compared with LSTM, GRU, and MLP models on 7 different Chinese cities showing smaller accuracy error w.r.t. all other models, especially MLP.

A method that exploits CNNs, and that has not been sufficiently explored yet in the field of predictive epidemiology, is Temporal CNN (TCN), which showed to have comparable, if not better, performance w.r.t. RNNs. The approach was first presented in [100] and it is based on the deployment of *causal* convolution for *sequence-to-sequence* learning tasks. In this method, the kernel is applied only over data relative to the present time step or previous input points in the sequence, avoiding information leakage from the future to the past. The network architecture is based on 1-D fully convolutional layers with the same length as the input data. This methodology is gaining increasing attention from the DL community and we regard it as a valuable approach going forward in the research of forecasting systems for epidemics.

Composed Deep Learning Models

In many instances, combining multiple DL-based methodologies can greatly improve the accuracy of the final predictions. For instance, integrating two models, one approximating the temporal features and the other encapsulating spatial characteristics can increase the overall performance. For this reason, a plethora of works integrating one or more DL approaches has been proposed in recent years.

Recursive Networks Autoencoders. An interesting set of DL-based methods, applied in the field of predictive epidemiology, deploy the so-called LSTM Autoencoders. In general, autoencoders are composed of two parts: an *encoder*, which creates an internal representation of the input sequences, namely the internal state of the LSTM cells used to model the sequence, and a *decoder*, which maps the embedding into the output feature space (i.e., a series of dense layer minimizing a reconstruction loss).

An application of this method is proposed in [1], where it has been deployed in the context of the FluSight Task[4], a challenge launched by the CDC to advance the research w.r.t. seasonal influenza forecasting. The proposed model is trained on the historical data relative to seasonal flu in the last 20 years. The authors implemented a system based on an LSTM paired with an attention mechanism that embeds the historical data into a latent space, whereas the decoder performs the actual predictions. The data relative to the trend of the current flu season is matched to the most similar historical series embedding through a Deep Clustering technique on the latent space. The historical epidemiological curve is reconstructed and used to predict the evolution of the ongoing outbreak of flu.

Another example, inspired by Variation Autoencoder, was proposed by [82]. The model, which is based on a variational LSTM-Autoencoder is composed of two branches trained in parallel. The first branch is a self-attention LSTM encoder, whose inputs are the daily new infected cases, along with government policies and *urban features* (population density, fertility rate, etc.). The second branch is an encoder, whose input is a distance-weighted adjacency matrix, computed using the geographical locations of the countries. The output of the encoder is a set of latent variables that are concatenated with the output of the first branch and passed to an LSTM which outputs the prediction of the daily cases per country.

[4]https://www.citizenscience.gov/catalog/171/

Combining Recursive and Convolutional Neural Networks. We have seen in the previous section how both RNNs and CNNs can be suitable choices to model the dynamics of a disease. An application combining both of these approaches is proposed in [143] with Convolutional LSTM (Conv-LSTM). The model consists of a Stacked-LSTM, where instead of traditional matrix multiplication, a convolutional operator is alternated for each layer. The model is compared to other RNN methods, including Bi-LSTM and Stacked-LSTM, showing improved accuracy.

On the same line, a method combining temporal information, i.e. 1D time series, with spatial information such as the disease dynamics of neighboring countries is introduced in [80]. The model combines the results of two different pipelines: the first considers as input the number of new daily recovered cases, deaths, confirmed cases, and the cumulative count of these features of the past 5 days. Such input is given to a 1D-CNN serially combined with a bi-GRU to extract temporal patterns from the input trends. The second pipeline, given the 2D matrix representing the epidemic trend of different regions in time, extracts the spatial correlation patterns through the use of a 2D-CNN. Such input matrix models the presence of epidemic trends of highly correlated and dependent neighboring countries. Ultimately, the outputs of both pipelines are concatenated, and a dropout layer is used to prevent overfitting in case of scarce data.

A similar method was proposed by [169] for the influenza epidemic. The researchers considered the data of the USA and Japan, i.e., the weekly reported patient count over a single state or prefecture. In this work, the authors used CNNs to fuse information from different data sources (e.g., different regions) and GRUs to capture the long-term correlation in the time series, with a residual structure. The residual structure is adopted to introduce highly relevant long jumps in the information flow (e.g., capture annual epidemiology patterns). We can observe that the CNNs were used to learn correlations *among* signals, while GRUs were targeting correlation *within* every single signal. However, the truly innovative solution provided in this work is the use of an adjacent nearest-neighbor matrix as a filter of the convolutional kernel of each country. This sort of adjacent convolution gets parameters focused on more local information in opposition to the classical grid convolution, implying that a single filter could be able to represent more complex patterns, which are usually captured by multiple filters when using the classical grid convolution.

Graph Neural Networks Geometric Deep Learning, and more specifically Graph Neural Networks (GNNs), has produced great results in many fields of research. Even if the use of GNNs is currently limited in computational epidemiology, we consider this as one of the most promising frameworks for virus spreading forecasting. Many of the traditional epidemiological models have deployed complex networks to describe the dynamic of epidemics. Consequently, GNNs are capable of extending the principle at the foundation to such data structure. For this reason, we reckon that a very promising direction consists of exploring GNNs that explicitly model the graph-based structure encoding the topological relations at the heart of epidemics spreading (e.g., interaction among people, geographical distributions, etc).

A variety of GNN applications to the problem was studied in [110], where multiple declinations of this approach were tested on a synthetic dataset. The main idea is to model the spreading starting from a network of individuals (i.e., the nodes) that can interact in specific ways (i.e., the edges). Then approximate the function mapping the vertex into a positive real number representing the count of infected nodes through *node regression*, assuming to have some initial contagions in the population. This method can be either based on pure GNNs or on propositional learners, where the first uses the adjacent matrix during the training of the model while the second does not. The node regression was ultimately implemented with either graph-based methods, such as *Graph Attention Networks* and *Graph Isomorphic Networks*, or a Gradient Boosting model, namely XGBoost.

Another example of the application of this class of methods is provided in [42] with Cola-GNN, a model aimed at producing reliable long-term epidemiological predictions. The Cola-GNN is one of the first original works to apply Graph Neural Networks in epidemic forecasting. The framework is composed of three main components: a *Location-aware Attention* element for capturing spatial local dependencies through an RNN, a *Temporal Convolution Layer*, in charge of approximating the temporal features, and a *Global Graph Neural Network*, that encapsulates the outputs of both previous layers and updates the nodes in a *message-passing* fashion, modeling the spreading of the virus.

Hybrid Deep Learning for Epidemics

A recent trend in the development of ANN-based methods relies on the integration of domain-specific knowledge [116]. The general idea is to design the learning process to incorporate data and techniques that are specifically referred to some particular aspect of the domain in consideration. In the case of predictive epidemiology, this tendency has materialized in hybrid approaches integrating compartmental models and deep learning. Historically, many forecasting methods for virus-spreading have been proposed [51]. Among the most notable we find compartmental models, i.e. mathematical models where the population is divided into categories representing different conditions w.r.t. the epidemic, and that describe the progression of the virus through differential equations (e.g. SIR: **S**usceptible-**I**nfected-**R**ecovered).

In [55] this idea is applied to predict the evolution of the COVID-19 epidemic in India. More specifically, the proposed framework is based on a SIRVD (**S**usceptible-**I**nfected-**R**ecovered-**V**accinated-**D**eceased) dynamic system which approximates the epidemiological curve; the parameters describing the transition into the different compartments are learned through an incremental learning approach. The idea is that a Feed-Forward Neural Network (FFNN) can be used to iteratively approximate the incoming data, which is updated based on the progression of the pandemic; this allows the improvement of the model without training the ANN on the whole dataset every time is updated. Similarly [86] proposed a *forward-inverse* neural network for SIR models. This solution can be considered as a sort of *end-to-end* approach where the parameters and the compartment (or category, such as the infected individuals in the population) at each time step are estimated with a time-dependent FFNN based on the historical data of COVID-19 in South Korea. This model shows the gain in terms of explainability deriving from the use of a hybrid framework.

Another model exploiting compartmental systems was introduced by [161]. In this case, the proposed methodology attempts to solve problems related to scarce data with a data augmentation framework for forecasting the incidence of Influenza-Like Illnesses (ILI) in the USA. The model, which is called Theory Guided Deep Learning Epidemic Forecasting with Synthetic Information, or TDEFSI, integrates the strengths of deep neural networks and high-resolution simulations of epidemic processes over complex networks. The data augmentation is performed by generating synthetic data using computer-based simulations of causal processes that capture epidemic dynamics: for instance, given a US state and an existing disease model (e.g., **S**usceptible-**E**xposed-**I**nfected-**R**ecovered), a marginal distribution is estimated for each parameter in the model. Then, a synthetic training dataset is generated for the target US state by sampling from the learned marginal distributions. The core model is composed of a two-branch LSTM-based deep neural network that captures the temporal dynamics of both within-season observations and between-season observations. The main advantage of this method is the capability to synthesize a large volume of time series from simulations based on epidemiology theory, reducing the risk of overfitting while training the model. At the same time, this method introduces the challenge of minimizing the gap between synthetic and real data, which is a non-trivial problem in itself.

Category	Sub-Category	Paper	Type	S	E	C	OD	MR	DA	X
Deep Learning	Simple	[142]	Bi-LSTM				✓	✓		
		[155]	LSTM	✓			✓	✓		✓
		[97]	CNN				✓	✓		
		[143]	Conv-LSTM				✓	✓		
		[79]	CNN				✓	✓		
	Composed	[1]	AE + LSTM + clustering			✓	✓			
		[143]	Conv-LSTM				✓	✓		
		[82]	VAE + LSTM	✓				✓		✓
		[80]	CNN + Bi-GRU	✓			✓	✓		
		[169]	CNN + GRU	✓		✓	✓	✓		
		[110]	GNN			✓				
		[42]	GNN + RNN + CNN	✓			✓	✓		
Hybrid Deep learning	Autoregressive	[24]	ARIMA + NNAR				✓	✓		
		[166]	ARIMA + NNAR				✓			
		[164]	SARIMA + NNAR				✓			
		[165]	SARIMA + NNARX				✓			✓
	Compartmental	[55]	FFNN + SIRVD		✓					
		[86]	FFNN + SIR		✓			✓		
		[161]	LSTM + SEIR					✓	✓	

Table 2.3: Features of the models: Spatial Data (S): spatial or geographical data ; Explainabile (E); Code available (C); Open Data (OD): data are publicly available; Multiple Countries/Regions (MR): the model considers multiple regions and countries; Data Augmentation (DA); Contextual/exogenous information (X): the model is integrated with input not directly related to the learning task, but that affect the outcome of the prediction.

2.3.3 Epidemiological Data

Deep Learning is a data-driven approach, therefore its performance relies heavily on the presence of accurate and consistent data. For this reason, the data-gathering strategy is of high relevance when approaching a learning task. In this view, the coronavirus pandemic has allowed to collect of a large amount of information, being the first global epidemic in an era marked by the presence of smartphones and big data systems.
In the scope of epidemics, we can distinguish two general types of data, which implicitly define different learning objectives: *clinical data*, describing medical aspects of the disease (e.g., hospital records, symptoms, administration of drugs), and *spatio-temporal data*, underlying the spreading of the virus, usually in terms of daily or weekly new infected cases. Among the main sources of information we can distinguish:

- **Government and health records**: governments, hospital administrations, and health organizations can voluntarily release records regarding citizens, either publicly, or in the context of a specific project.
- **Mobile data**: the increasing presence of smartphones in our lives

has made it easier to gather information about our habits, individual movements, and contact patterns [121], producing valuable information for both epidemic control and virus-spreading forecasting.

- **Social media and search engines**: along with the growing use of smartphones, social media, and search engines can be great sources of information, ranging from reported symptoms, inferred from public posts, to search queries. One interesting example is shown in [137], where Twitter messages and web queries were used to estimate the flu trend.

- **Sensors**: information can also be collected through the use of specific pieces of hardware, or sensors, as shown in [138], where Radio-Frequency-Identification (RFID) technology is used to collect data regarding the contacts among individuals in a small environment, and then recreate a possible chain of infections.

The presence of a data-gathering infrastructure has improved the quantity and quality of data available to the research. However, there are many issues concerning the possible consequences on human privacy and personal freedom, with an increasing demand for stricter rules and a secure design of these systems [147].

Ultimately, the quality of DL models depends on the amount of data available, but, unfortunately, information can be scarce, especially during the first stages of the epidemic. Some authors have specifically addressed this issue with data augmentation methodologies, as proposed in [161], or smart strategies to exploit historical data [1, 122].

Chapter 3

Tree-based Data Analysis for Infection Disease

A notable attribute of machine learning models and statistical analyses lies in their ability to discern pertinent features that substantially influence the outcome of predictive tasks. This process of feature identification yields valuable insights, enriching our comprehension of specific phenomena. In the realm of healthcare, this capacity is particularly valuable, as it empowers domain experts to extract knowledge from data, thereby facilitating the formulation of novel hypotheses and therapeutic approaches. Traditionally, this challenge has been addressed using linear techniques such as Linear and Logistic Regressions. Nonetheless, these methods, while effective, often grapple with the limiting assumption of linearity in the relationships between variables. Our objective in this analysis is to enhance the analytical capabilities available to healthcare professionals. We accomplish this by harnessing decision tree-based solutions (as detailed in Section 2.1.1), which exhibit the ability to approximate intricate relationships. Specifically, these methods are suited for capturing non-linear correlations within the data. Nevertheless, the accuracy of Decision Trees and Random Forests is closely intertwined with data availability and quality. Indeed, on one hand, acquiring data about medical patients can be challenging, as they are safeguarded by the GDPR, while on the other hand, data may often exhibit spurious correlations, which can impede the overall quality of the analysis.

This work addresses these issues directly by proposing a data augmentation and decorrelation methodology, aiming to provide robust and reliable analyses of medical records. In the first part of this chapter, we will describe the primary methodologies applied, while in the second part, we will apply this approach to a specific case study.

3.1 Feature Importance in Tree-based models

Tree-based machine learning methods provide a powerful tool to quantify the importance of each input attribute in the prediction process. This mechanism is rooted in the way individual Decision Trees (DTs) function. When a tree model is trained and a split is created, it is associated with an increase in a performance indicator (e.g., Gini Index, Entropy). By aggregating these increases across all attribute splits, we can estimate their impact on the tree's behavior. These estimates are known as feature importance scores and can be computed for any tree-based machine learning method, including Random Forest. In the context of our analysis, feature importance scores are the key element to produce explaination to the phanomenon under scrutiny. It's important to note that, since most machine learning models operate identifying correlations in the data, a high feature importance score does not necessarily indicate a cause-and-effect relationship. Furthermore, it's essential to recognize that the reliability of feature importance scores is contingent on the performance of the machine learning model itself. In other words, the importance scores derived from an inaccurate model can be misleading.

3.2 Feature Selection with Boruta

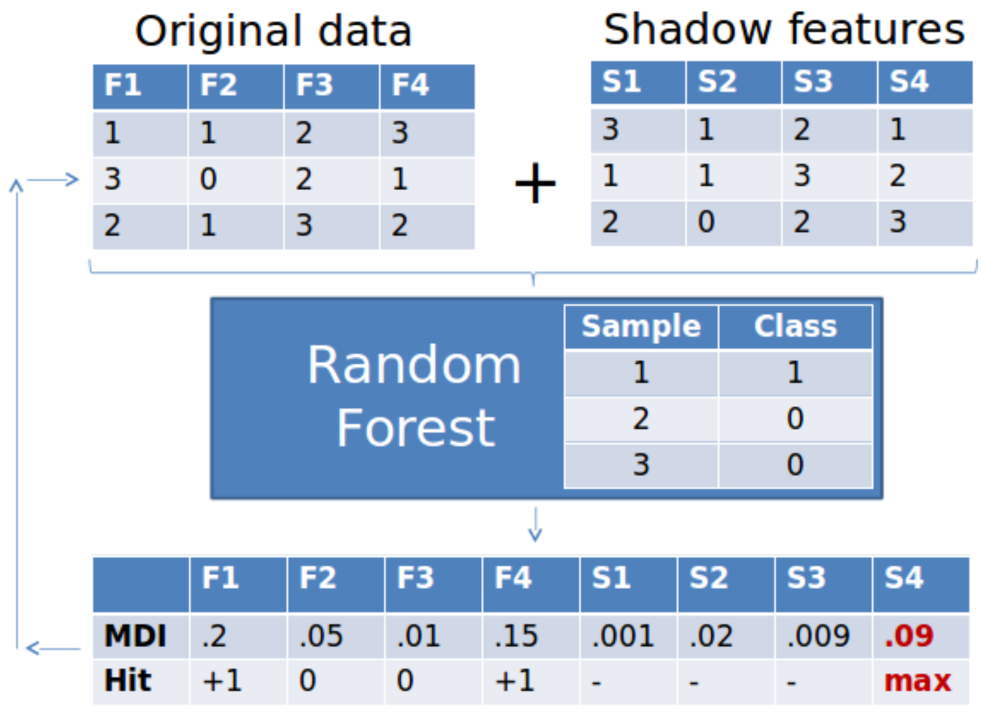
Tree-based methods assume that the dataset does not contain highly correlated variables, as such correlations could potentially distort the feature importance analysis. When two variables are strongly correlated, they essentially provide similar information gain to the tree and can be used interchangeably. However, achieving this condition is not always feasible, especially when dealing with high-dimensional data, such as extensive clinical records with numerous features. To address this challenge, a popular solution is feature selection. Feature selection enables the model to focus on the most important features or those with statistically significant importance. This approach reduces the risk of overfitting and yields more reliable importance scores as a byproduct. A straightforward approach for Tree-based methods is to use the Information Gain value as a selection criterion. The idea is to establish a threshold of importance that determines which features are considered unimportant. However, this method does not directly address the presence of spurious correlations in the data, which can arise in high-dimensional datasets due to the presence of numerous variables with similar domains. An effective heuristic solution to this issue is provided by the Boruta framework. The underlying idea is to create variables that are randomly correlated with the outcome, making them appear unimportant. Essentially, any random permutation of attribute values is, by design, randomly correlated with the output. These newly generated randomly correlated features called *Shadow Features*, are then used as a selection criterion. The assumption here is that any variable less important than a randomly correlated one is, indeed, unimportant. The method underlying Boruta is shown in Algorithm 1 and Figure 3.1.

Algorithm 1 Boruta Algorithm

```
1: procedure BORUTA(Dataset)
2:     Input: Dataset with $m$ features
3:     Output: Set of relevant features
4:     Duplicate dataset creating shadow attributes for each feature
5:     Shuffle entries of shadow attributes
6:     repeat
7:         Train a random forest classifier on the augmented dataset
8:         Record importance of each attribute $Z_i$
9:         $Z_{\text{max\_shadow}} \leftarrow$ max(importance of shadow features)
10:        for each original feature $i$ do
11:            if $Z_i \geq Z_{\text{max\_shadow}}$ then
12:                Mark feature $i$ as relevant
13:            else
14:                Mark feature $i$ as unimportant
15:            end if
16:        end for
17:    until all features are either marked relevant or unimportant or a
       predefined iteration limit is reached
18:    return set of relevant features
19: end procedure
```

3.3 Data Augmentation

As mentioned earlier, the accuracy of an ML model is greatly reliant on the amount of data available for the training process. A larger dataset allows for a more precise approximation and better generalization of the underlying distribution related to the phenomenon under analysis. To mitigate



F1	F2	F3	F4
1	1	2	3
3	0	2	1
2	1	3	2

S1	S2	S3	S4
3	1	2	1
1	1	3	2
2	0	2	3

Sample	Class
1	1
2	0
3	0

	F1	F2	F3	F4	S1	S2	S3	S4
MDI	.2	.05	.01	.15	.001	.02	.009	.09
Hit	+1	0	0	+1	-	-	-	max

Figure 3.1: Boruta Framework - source: danielhomola

this limitation, we employed a method of data enrichment or augmentation, as detailed in Section 2.2.3. This technique involves transforming the feature space to generate additional data samples. Essentially, we reframed the problem to create a new one that preserves similar properties to the original. This transformation was executed by considering pairwise comparisons between instances, where each new data entry is calculated as the ratio between the variables of two entries in the dataset. For example, if each entry in the dataset represents a patient's record, the transformed data might represent relative differences between two patients Consequently, the number of data points increases quadratically, transitioning from n samples to $\sum_i n - i$. It is crucial to acknowledge that this process distorts the feature space.

3.4 Decorrelation

Feature importance enables the identification of the most influential features correlated with a specific outcome. However, in a medical context, some variables considered important might merely be *trivial explainers*, or what adopting a causal perspective could be considered a *confounder variable*. For

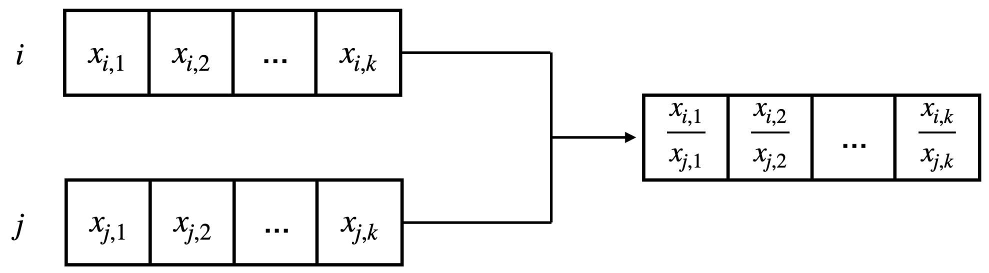

i
$x_{i,1}$
$x_{i,2}$
...
$x_{i,k}$
j
$x_{j,1}$
$x_{j,2}$
...
$x_{j,k}$
$\frac{x_{i,1}}{x_{j,1}}$
$\frac{x_{i,2}}{x_{j,2}}$
...
$\frac{x_{i,k}}{x_{j,k}}$

Figure 3.2: New instance computed from the features of patients i and j

instance, the chronological age of a patient when assessing the severity of a condition related to a specific disease might seem crucial. However, its actual contribution to the analysis could be considered as simplistic, especially given that older patients generally possess weaker defense mechanisms. A straightforward solution would be to exclude such variables from the dataset. Yet, this neglects potential correlations between the trivial explainer and other variables, possibly acting as proxies, or intermediary, for the confounder. For example, serum creatinine concentration tends to decrease with age [17], thus it might represent a substitute for chronological age in the analysis. A potential solution to this challenge involves adjusting, or mitigating, the impact of trivial explainers on input attributes and then removing them from the set of features considered. This can be accomplished by computing a discount factor for each of the variables involved. Specifically, we use the confounder as a predictor for the other features, e.g., a Neural Network or a Random Forest. The values predicted by the model can then serve as a normalization (discount) factor. This approach can lead to two scenarios:

- If the variable to be removed is a strong predictor of a feature, all its values will tend to a fixed value, diminishing its informativeness during the training process.
- Otherwise, the values of the variables will be rescaled based on the predicted value.

By implementing this method, we can better manage the impact of less relevant variables while retaining the full set of features in the analysis.

Experimental Results

Here, we showcase some of the results obtained by applying the methods presented in the previous section to a collection of clinical records regarding COVID-19 patients. In doing so, we focus on the use of Random Forest to provide the importance scores relative to the health conditions of each patient. All the main findings will be presented outlining the advantages and disadvantages of each approach. The results of our analysis are visually represented in Figures 3.3, 3.4, and 3.5. Additionally, the accuracy metrics are comprehensively tabulated in Table 3.2. It is important to note that in the feature importance plots, we have restricted the variables displayed to the top 10 most critical features. Moreover, to ensure the robustness and reliability of our findings, we employed a K-fold cross-validation technique, setting K equal to 10. This allowed us to equip the results presented in Table 3.2 with a standard deviation, providing insight into the stability of the pipeline.

The following analysis aims to identify key factors in high-risk individuals affected by SARS-CoV-2. Specifically, we considered data from 5645 subjects seen at Hospital Israelita Albert Einstein in São Paulo, Brazil. Our analytical focus is centered on predicting the likelihood of ICU admission, a criterion we have adopted as a proxy for gauging the severity of the illness. The data used in this analysis are publicly available on Kaggle[1]. To initiate our analysis, we engaged in preprocessing the data to render it amenable to the machine learning model. In doing so, we took the following steps:

- Removed variables with several missing values exceeding the 30% threshold.
- Removed patients' records with several missing values exceeding the 30% threshold.
- Processed categorical data by introducing numerical encoding.

During the preprocessing phase, the number of variables considered was reduced from 111 to 36, which are highlighted in Table 3.1, while the sample of patients considered decreased from 5645 to 366.

Feature Selection

The application of Boruta has narrowed down the set of variables to 15 candidates. Among these, the predictor with the highest importance is the

[1]https://www.kaggle.com/datasets/einsteindata4u/covid19

Patient age quantile	Hematocrit
Hemoglobin	Platelets
Mean platelet volume	Red blood Cells
Lymphocytes	MCV concentration (MCHC)
Leukocytes	Basophils
Mean corpuscular hemoglobin (MCH)	Eosinophils
Mean corpuscular volume (MCV)	Monocytes
Red blood cell distribution width (RDW)	Respiratory Syncytial Virus
Influenza A	Influenza B
Parainfluenza 1	CoronavirusNL63
Rhinovirus/Enterovirus	Coronavirus HKU1
Parainfluenza 3	Chlamydophila pneumoniae
Adenovirus	Parainfluenza 4
Coronavirus229E	CoronavirusOC43
Inf A H1N1 2009	Bordetella pertussis
Metapneumovirus	Parainfluenza 2
Influenza B, rapid test	Influenza A, rapid test
Proteina C reativa mg/dL	

Table 3.1: List of Features

Model / Method	Accuracy	AUC
FS	0.88 ± 0.012	0.81 ± 0.12
FS + AUG	0.91 ± 0.0037	0.80 ± 0.060
FS + AUG + DEC	0.91 ± 0.022	0.76 ± 0.090

Table 3.2: Evaluation Metrics Feature Importance: Accuracy and AUC; Feature Selectin (FS); Data Augmentation (AUG); Age Decorrelation (DEC)

C reactive protein, as illustrated in Figure 3.3. As reported in Table 3.2, this approach demonstrates a fairly accurate performance, with an AUC value consistent with expected levels. However, there is potential for further improvement; applying the data augmentation method in Section 3.3 should provide a more stable and robust approximation.

Feature Selection and Data Augmentation

In this step, we shifted the focus of our model's underlying question. Instead of asking, *Given information about subjects, what is the likelihood of developing a severe form of the disease?* we reformulated the question to, *Given information about the ratios of input attributes for a pair of subjects, which*

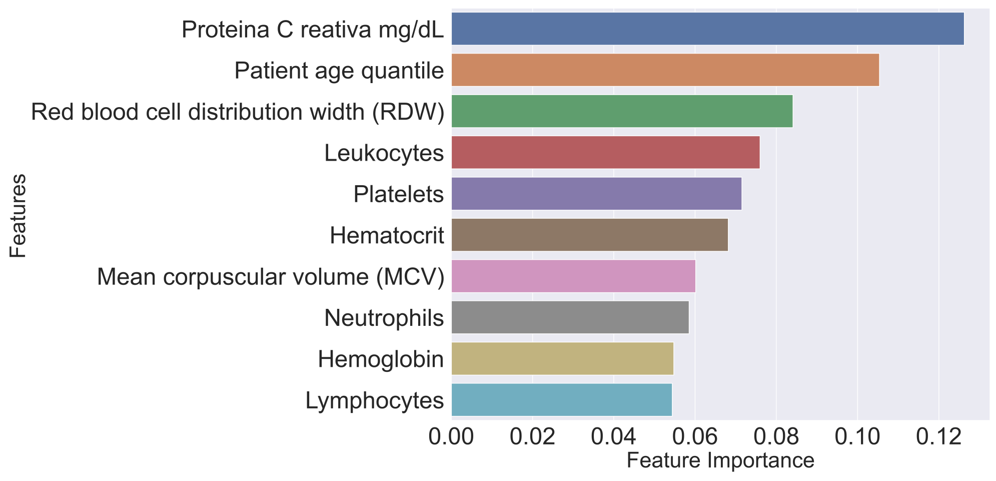


Figure 3.3: Feature Importance of Random Forest with Feature Selection

subject has a higher risk? Subsequently, our target variable, a categorical attribute indicating whether a patient was hospitalized in the ICU (1) or not (0), can be redefined as follows:

- If the first patient in the comparison has been hospitalized in the ICU, while the second has not, it implies a more severe form of the illness.
- If not, it suggests a form of the illness of at least similar severity to the other patient.

In scientific terms, the new question is less stringent than the previous one. In fact, with an oracle for the former question, it is possible to answer the latter, but the reverse is not true. That being said, the new question remains highly informative within the context of our main objective. For example, if the ratio between attributes like Hematocrit for two subjects can predict the likelihood of being hospitalized in the ICU of the first patient, it provides valuable insights for our analysis. As expected the accuracy of the model increases through the use of data augmentation. This is consistent with the objective of the general approach. We can also observe comparing Figure 3.3 and Figure 3.4, that the ranking of variables based on the importance score is preserved.

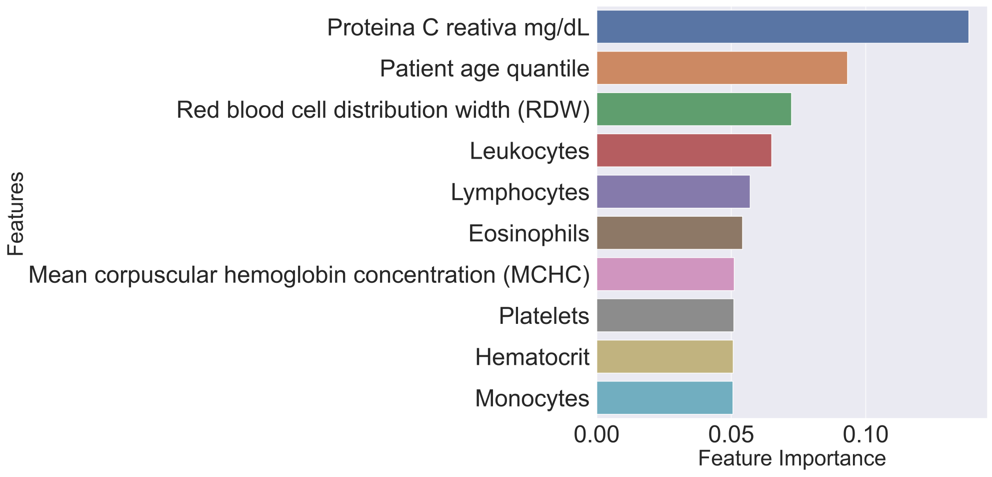


Figure 3.4: Feature Importance of Random Forest with Feature Selection

Feature Selection, Data Augmentation and Decorrelation

The previous analysis highlighted the importance of a patient's age as one of the most influential factors in determining the severity of the illness. However, it is important to note that these pieces of information may be considered obvious: age is often linked to increased vulnerability. Hence, one could argue that this is a self-evident detail or a confounding element for our ultimate goal. To diminish the impact of this feature and facilitate a more insightful examination, we removed it through the decorrelation method explained in Section 3.4. Thus, the chronological age dropped from our dataset, and each variable in rescaled based on a discount factor computed using a shallow neural network.

The results in Figure 3.5 show that the decorrelation process effectively changed the feature importance. Most notably, we can observe a decreased importance of the RDW factor, which is linked to the age of the subject [76]. We can also observe how Leukocytes are now deemed unimportant and excluded by the most impactful features.

Discussion

The results show that ML algorithms can be applied with high accuracy to identify variables likely involved in the patient's conditions. Considering all the steps described above (tree-based models, feature importance, Boruta, data augmentation, decorrelation) we have provided a well-performing model

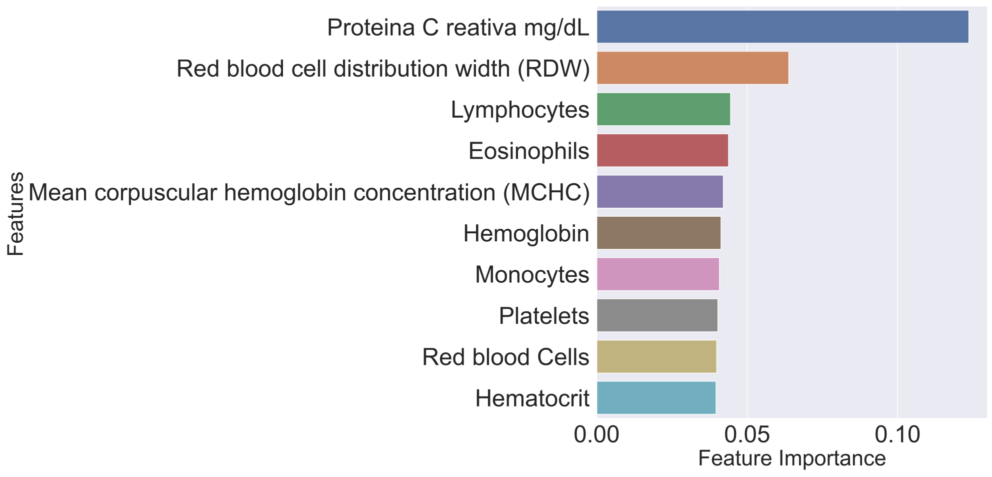


Figure 3.5: Feature Importance of Random Forest with Feature Selection, Data Augmentation and Decorrelation

with high accuracy, as shown in Table 3.2. During the analysis, we have identified as variables of interest:

- **C-reactive protein**: Often considered a reliable biomarker for detecting inflammation in the body. Recent studies related to COVID-19 have shown that it can serve as an early marker indicating the severity of the illness in COVID-19 patients [2]. Moreover, it is frequently associated with the mortality of subjects [148]

- **Red blood cell distribution width (RDW)**: Often associated with poor health conditions, especially when elevated values are observed. In patients affected by COVID-19, this factor has been frequently found in subjects with a higher risk of in-hospital mortality and septic shock [76].

- **Hemoglobin** and **MCHC**: These variables align with the results of studies indicating prolonged alterations in hematological parameters and hemoglobin concentration in patients affected by Sars-CoV-2 [68, 49].

This study underscores the vast potential of machine learning (ML) solutions in analyzing clinical records. It enables the identification of non-linear relationships within the sample, offering a broader and more insightful perspective on novel findings. Furthermore, ML methods can be integrated with various techniques, fostering a higher level of accuracy and facilitating more robust and precise knowledge extraction. Nevertheless, we want to outline the limitations and suggest further improvements that could enhance the methods presented in this paper:

- It is important to emphasize that Boruta is a heuristic method for feature selection and does not guarantee the elimination of all correlations in the dataset.

- The data augmentation process, as formulated in the scope of this study, represents a distortion of the original feature space. Thus, there is no guarantee of the preservation of specific properties, e.g., monotonicity.

- The decorrelation method emerges as a highly promising tool, warranting further exploration and investigation. More importantly, we performed decorrelation concerning a single variable, namely the chronological age. However, there is potential to extend this approach by removing additional variables that are considered trivial or non-contributory to the patient's condition. This could be a process informed by the knowledge provided by domain experts that could point out well-known confounders.

As a final remark, we want to highlight that the methodologies presented in this chapter have been successfully applied in other studies, with promising results [10].

Chapter 4

Informed Deep Learning for Epidemic Forecasting

In Section 2.3.2, we outlined a taxonomy of deep learning methods for epidemic forecasting. The key limitation of these methods is their heavy reliance on data availability, which can severely impact their accuracy. This problem is more evident during the early stages of epidemics when data are scarce, if not absent.
In this chapter, we introduce two methods aimed at addressing these challenges. Both the DL models rely on the injection of prior knowledge, which in this case takes the form of compartmental models and renewal equations (Section 2.3.1). To this end, the state-of-the-art analysis provided in Section 2.2.1 is crucial in understanding how we approached the problem.
The rest of the chapter is articulated into these main topics:

- **Universal Ordinary Differential Equations**: this approach falls under the category of methods discussed in Section 2.2.4, often referred to as Physics-Informed Neural Networks (PINNs). In the context of epidemics, it can be combined with compartmental models to predict epidemic trends.

- **Deep Renewal Equations**: which exploit Renewal Equations (Section 2.3.1) as a mathematical framework to guide the learning process of a deep learning model.

- **Transfer Learning**: that extends the methods above to provide a robust and general solution, opening to 0-shot approximation of epidemiological phenomenon.

4.1 Universal Ordinary Differential Equation

This section provides a thorough analysis of a specific approach known as Universal Differential Equation (UODE) [126]. The UODE framework offers a highly generalizable description of many classical Physics-Informed Neural Network (PINN) models. In the initial part of this section, we extensively examine the potential and limitations of this method, while in the latter part, we delve deeper into its application for epidemic forecasting.

4.1.1 Universal Differential Equations for data-driven discovery of ODEs

Purely data-driven methods, like Deep Neural Networks (DNNs), have huge representational power and can deal with noisy high dimensional raw data; however, they may learn observational biases, leading to physically inconsistent predictions and poor generalization performance. On the other hand, despite the relentless progress in the field, solving real-world partial differential equations (PDEs) using traditional analytical or computational approaches requires complex formulations and prohibitive costs. A lot of effort has been devoted to bridging DNNs with differential equations in end-to-end trainable frameworks. However, less attention has been paid to analyzing the advantages and limitations of the proposed approaches.

We view the lack of an in-depth analysis of physics-informed techniques as a major issue. We contribute to this area by performing an analysis on the Universal Differential Equation (UDE) [126] framework in the context of data-driven discovery of ODEs. We focus on UDE since its general formulation allows us to express other existing frameworks. In particular, we focus on: 1) *evaluating two training approaches in terms of accuracy and efficiency*; 2) *testing the effect of the numerical solver accuracy in the parameters approximation*, 3) *analyzing the impact of the data collection process regarding the approximation accuracy*, and in 4) *exploring the effectiveness of UDE in reconstructing a functional dependence between a set of observables and the unknown parameters.*

Universal Differential Equations. The UODE formulation relies on embedded universal approximators to model forced stochastic delay PDEs in the form:

$$\mathcal{N}\left[u(t), u(\alpha(t)), \mathrm{W}(t), U_\theta(u, \beta(t))\right] = 0 \tag{4.1}$$

where $u(t)$ is the system state at time t, $\alpha(t)$ is a delay function, and $\mathrm{W}(t)$ is the Wiener process. $\mathcal{N}[\cdot]$ is a nonlinear operator and $U_\theta(\cdot)$ is a universal approximator parameterized by θ. The UODE framework is general enough to express other frameworks that combine physics knowledge and deep learning models. For example, by considering a one-dimensional UODE defined by a neural network, namely $u' = U_\theta(u(t), t)$, we retrieve the Neural Ordinary Differential Equation framework

UODEs are trained by minimizing a cost function C_θ defined on the current solution $u_\theta(t)$ with respect to the parameters θ. The cost function is usually computed on discrete data points (t_i, y_i) which represent a set of measurements of the system state, and the optimization can be achieved via gradient-based methods like ADAM [95] or Stochastic Gradient Descent (SGD) [131].

In this section, we present the UDE formulation we adopt, and we describe four research questions aimed at performing an in-depth analysis of the UDEs framework in solving a data-driven discovery of ODEs.

Formulation

We restrict our analysis to dynamical systems described by ODEs with no stochasticity or time delay. The corresponding UDE formulation is:

$$u' = f(u(t), t, U_\theta(u(t), t)). \tag{4.2}$$

where $f(\cdot)$ is the known dynamics of the system, and $U_\theta(\cdot,\cdot)$ is the universal approximator for the unknown parameters. As cost function, we adopt the Mean Squared Error (MSE) between the current approximate solution $u_\theta(t)$ and the true measurement $y(t)$, formally:

$$C_\theta = \sum_i \|u_\theta(t_i) - y(t_i)\|_2^2 \,. \tag{4.3}$$

We consider discrete time models, where the differential equation in (4.2) can be solved via numerical techniques. Among the available solvers, we rely on the Euler method, which is fully differentiable and allows for gradient-based optimization. Moreover, the limited accuracy of this first-order method enlightens the effects of the integration technique on the unknown parameter approximation. Our analysis starts from a simplified setting, in which we assume that the unknown parameters are fixed. Therefore, the universal approximator in 4.2 reduces to a set of learnable variables, leading to:

$$u' = f(u(t), t, U_\theta). \tag{4.4}$$

Training Procedure. Given a set of state measurements y in the discrete interval $[t_0, t_n]$, we consider two approaches to learn Equation (4.4), which we analyze in terms of *accuracy* and *efficiency*. The first approach, mentioned by [128] and named here FULL-BATCH, involves 1) applying the Euler method on the whole temporal series with $y(t_0)$ as the initial condition, 2) computing the cost function C_θ, and 3) optimizing the parameters θ via full-batch gradient-based methods. An alternative approach, named MINI-BATCH, consists of splitting the dataset into pairs of consecutive measurements $(y(t_i), y(t_{i+1}))$, and considering each pair as a single initial value problem. Then, by applying the Euler method on the single pair, we can perform a mini-batch training procedure, which helps in mitigating the gradient vanishing problem [64]. Conversely to the FULL-BATCH approach, which requires data to be ordered and uniform in observations, the MINI-BATCH method has less strict requirements and can be applied also to partially ordered datasets.

Solver Accuracy. In the UDE framework, the model is trained to correctly predict the system evolution by learning an approximation of the unknown parameters that minimize the cost function C_θ. The formulation relies on the integration method to approximate the system state $u(t)$. However, the numerical solver may introduce approximation errors that affect the whole learning procedure. Here, we want to investigate the *impact of the solver accuracy on the unknown parameters approximation.* Since the Euler method is a first-order method, its error depends on the number of iterations per time step used to estimate the value of the integral, and, thus, we can perform our analysis with direct control on the trade-off between execution time and solver accuracy.

Functional Dependence. By relying on the universal approximator in 4.2, the UDE framework can learn not only fixed values for the unknown parameters but also functional relationships between them and the observable variables. Thus, we add a level of complexity to our analysis by considering the system parameters as functions of observable variables, and we *evaluate the UDE accuracy in approximating the unknown functional dependence.*

Data Sampling. Since UDE framework is a data-driven approach, it is important to investigate the effectiveness of the UDE framework under different data samplings. In particular, *can we use the known dynamics of the system under analysis to design the data collection process to increase the approximation accuracy?*

Empirical Analysis

Here, we report the results of our analysis performed on two case studies: 1) *RC circuit*, i.e., estimating the final voltage in a first-order resistor-capacitor circuit; 2) *Predictive Epidemiology*, i.e., predicting the number of infected people during a pandemic. We start by describing the two case studies; then, we illustrate the evaluation procedure and the experimental setup. Finally, we present the experiments focused on the research questions highlighted in Section 4.1.1, and we report the corresponding results.

RC Circuit. We consider a first-order RC circuit with a constant voltage generator. The state evolution of the system is described by

$$\frac{dV_C(t)}{dt} = \frac{1}{\tau}(V_s - V_C(t)) \tag{4.5}$$

where $V_C(t)$ is the capacitor voltage at time t, V_s is the voltage provided by the generator, and τ is the time constant which defines the circuit response.

We use the UDE formulation to approximate τ and V_s by writing Equation (4.5) as

$$u' = \frac{1}{U_{\theta_1}(t)}(U_{\theta_2}(t) - u(t)) \tag{4.6}$$

where u_t is a short notation for $V_C(t)$, $U_{\theta_1}(t)$ and $U_{\theta_2}(t)$ are the neural networks approximating τ and V_s respectively. The cost function is defined as

$$C_{\theta_1,\theta_2} = \sum_i (u_{\theta_1,\theta_2}(t_i) - y_i)^2 \tag{4.7}$$

where $u_{\theta_1,\theta_2}(t_i)$ and y_i are the current solution and the discrete-time measurements of the capacitor voltage at time t_i, respectively.

Predictive Epidemiology. Among the different compartmental models used to describe epidemics, we consider the well-known Susceptible-Infected-Recovered (SIR) model (Section 2.3.1), where the disease spreads through the interaction between susceptible and infected populations. The dynamics of a SIR model is described by the following set of differential equations:

$$\begin{aligned}
\frac{dS}{dt} &= -\beta\,\frac{S \cdot I}{N}, \\
\frac{dI}{dt} &= \beta\,\frac{S \cdot I}{N} - \gamma\, I, \\
\frac{dR}{dt} &= \gamma\, I,
\end{aligned} \tag{4.8}$$

where S, I, and R refer to the number of susceptible, infected, and recovered individuals in the population. The population is fixed, so $N = S + I + R$. The parameter $\gamma \in [0, 1]$ depends on the average recovery time of an infected subject, while $\beta \in [0, 1]$ is the number of contacts needed per time steps to have a new infected in the susceptible population. β determines the spreading coefficient of the epidemic and is strongly affected by different environmental factors (e.g., temperature, population density, contact rate, etc.). The introduction of public health measures that directly intervene in these environmental factors allows to contain the epidemic spreading.

We rely on the UDE framework to i) perform system identification on a simulated SIR model, and ii) estimate the impact of *Non-Pharmaceutical Interventions* (NPIs) on the epidemic spreading. We define the state of the system at time t as $\mathbf{u}_t = (S_t, I_t, R_t)$ and we formulate the Equations in (4.8) as

$$\mathbf{u}' = f(\mathbf{u}_t, t, U_\theta(\mathbf{u}_t, t, X_t)) \tag{4.9}$$

where X_t is the set of NPIs applied at time t. We assume γ to be fixed and known, and we approximate the SIR model parameter β with a neural network $U_\theta(\mathbf{u}_t, t, X_t)$. The cost function for this case study is defined as

$$C_\theta = \sum_i (u_\theta(t_i) - \hat{y}_i)^2 \tag{4.10}$$

where $u_\theta(t_i)$ and y_i are the current solution and the discrete-time measurements of the system state at time t_i, respectively.

Evaluation and experimental setup.

We evaluate the model accuracy by relying on two metrics: the *Absolute Error* (AE), to evaluate the estimation of the parameters, and the *Root Mean Squared Error* (RMSE), to study the approximation of the state of the dynamic systems. For each experiment, we perform 100 trials, normalize the results, and report mean and standard deviation. The source code is available at https://github.com/ai-research-disi/ode-discovery-with-ude.

Training Procedure

We compare FULL-BATCH and MINI-BATCH methods to assess which is the most accurate and efficient. We rely on high-precision simulation to generate data for both case studies. For the RC circuit, we set $V_c(0) = 0$, and we sample 100 values of V_s and τ in the range $[5, 10]$ and $[2, 6]$, respectively. Then, we generate data by relying on the analytical solution of Equation

Table 4.1: Comparison between MINI-BATCH and FULL-BATCH methods in RC circuit use case. We report the AE of V_s and τ approximation, RMSE for $V_c(t)$ prediction, and computational time in seconds.

	V_s	τ	$V_c(t)$	Time
MINI-BATCH	0.027 ± 0.013	0.163 ± 0.101	0.021 ± 0.010	9.21 ± 39.49
FULL-BATCH	0.018 ± 0.021	0.200 ± 0.081	0.014 ± 0.020	26.19 ± 5.69

4.5. From each of the resulting curves, we sample 10 data points $(V_c(t), t)$ equally spaced in the temporal interval $[0, 5\tau]$. Concerning the epidemic case study, the data generation process relies on a highly accurate Euler integration with 10.000 iterations per time step. We use the same initial condition across all instances, namely 99% of susceptible and 1% of infected on the entire population, and we assume γ to be equal to 0.1, meaning that the recovery time of infected individuals is on average 10 days. We create 100 epidemic curves, each of them determined by the sampled value of β in the interval $[0.2, 0.4]$. The resulting curves contain daily data points in the temporal interval from day 0 to day 100 of the outbreak evolution.

We evaluate the accuracy of UDE in approximating the unknown parameters and the system state, and we keep track of the total computation time required to reach convergence. We believe it is relevant to specify that the MINI-BATCH has an advantage compared to the FULL-BATCH. The evaluation of the latter involves predicting the the whole state evolution given only the initial one u_0; whereas, the first approach reconstructs the state evolution if provided with intermediate values. Thus, to have a fair comparison, the predictions of the MINI-BATCH method are fed back to the model to forecast the entire temporal series given only u_0. As shown in 4.1, for the RC circuit case study, both FULL-BATCH and MINI-BATCH approximate quite accurately V_s and $V_c(t)$, whereas the approximation of τ has a non-negligible error. However, FULL-BATCH requires almost 3 times the computational time to converge. In the SIR use case (4.2), the two training procedures achieve very similar accuracies, but FULL-BATCH is more than 8 times computationally expensive. Since both the methods have very similar estimation accuracy, we can conclude that MINI-BATCH *is a more efficient method to train the UDE compared to* FULL-BATCH. Thus, we rely on the MINI-BATCH method in the remaining experiments.

Table 4.2: Comparison between MINI-BATCH and FULL-BATCH methods in epidemic use case. We report the AE of β approximation, RMSE for $SIR(t)$ prediction, and computational time in seconds.

	β	$SIR(t)$	Time
MINI-BATCH	0.0030 ± 0.0019	0.017 ± 0.0046	1.28 ± 0.23
FULL-BATCH	0.0065 ± 0.0053	0.019 ± 0.0079	10.23 ± 2.50

Solver Accuracy

In the context of ODE discovery, we are interested in approximating the unknown system parameters. Despite an overall accurate estimation of the system state, the results of the previous analysis show that UDE framework does not reach high accuracy in approximating the system parameters. The model inaccuracy might be caused by the approximation error introduced by the integration method. Thus, to investigate the *impact of the solver accuracy on the unknown parameters approximation*, we test different levels of solver accuracy by increasing the number of iterations between time steps in the integration process. A higher number of iterations per time step of the Euler method should lead to more accurate solutions of the ODE; however, this comes also at the cost of a higher computational time as shown in Figure 4.1.

In this experiment, we use the same data generated for *Training procedure* experiment. In 4.2, we report the approximation error of the UDE framework when applying the Euler method with an increasing number of steps. As expected, in both use cases, by increasing the precision of the Euler method, the ODE parameters estimation becomes more accurate, until reaching a plateau after 10 iterations per time step.

Functional Dependence and Data Sampling

In a real-world scenario, the dynamical systems that we are analyzing often depend on a set of external variables, or *observables*, that influence the behavior of the system. These elements can be environmental conditions or control variables which affect the evolution of the system state. We study the UDE framework in presence of observables, assuming two kinds of relationship between the independent and dependent variable, namely, a *linear* and a *non-linear* dependence.

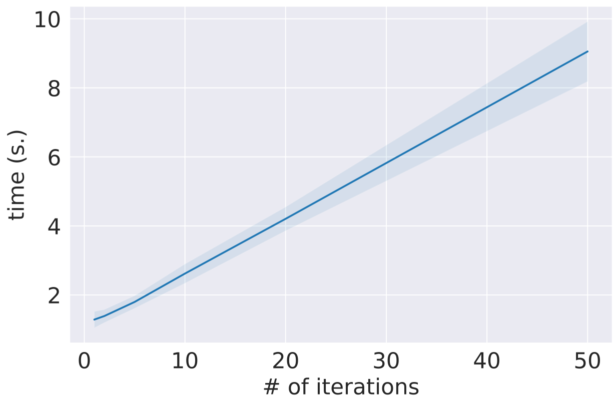
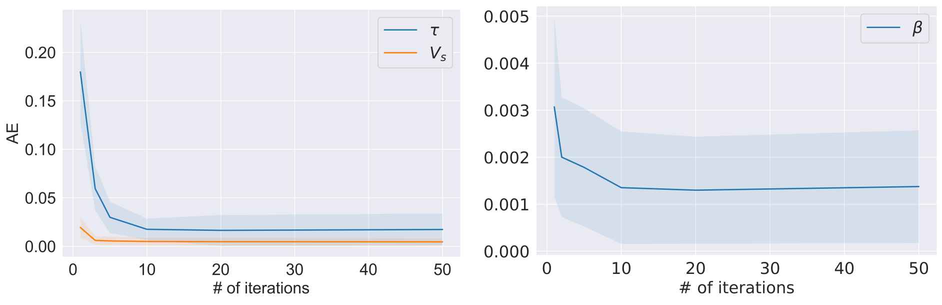


Figure 4.1: UDE training time as a function of the number of iterations per time step of the Euler method.



(a) τ and V_s (b) β

Figure 4.2: Average and standard deviation of the AE as a function of the number iterations per time step of the Euler method.

Linear Dependence For the RC circuit, we consider a simple and controlled setup where τ is a linear function of a continuous input variable x changing over time, namely $\tau(x) = ax$, where a and x are scalar values. Conversely, to the previous experiments, we assume V_s to be known and equal to 1; we perform this design choice to focus our analysis on the approximation accuracy of the linear relationship solely. Since the value of τ changes over time, we can not rely on the analytic solution of Equation (4.5) to generate data. Thus, we generate samples from one timestep to the successive one by running a high-resolution integration method, namely the Euler method with 10, 000 iterations per time step. In the generation process, the linear coefficient a is randomly sampled from a uniform probability distribution in the interval $[2, 6]$, and the observable x is initialized to 1, and updated at each time step as follows:

$$x(t) = x(t-1) + \epsilon, \quad \text{with} \quad \epsilon \sim \mathcal{U}_{[0,1]}.$$

This procedure allows to have reasonable variations of τ to prevent physically implausible data. During the learning process, as a consequence of the results obtained in the *Solver Accuracy* experiment (4.1.1), we use 10 iterations per time step in the Euler method as a trade-off between numerical error and computational efficiency.

In this experiment, we are interested in *evaluating the UDE accuracy in approximating the unknown linear dependence.* The resulting absolute error of the approximation of the linear coefficient a is 0.24 ± 0.27. Since the UDE is a data-driven approach, the estimation error may be due to the data quality. Since we simulate the RC circuit using a highly accurate integration method resolution, we can assume that data points are not affected by noise. However, the sampling procedure may have a relevant impact on the learning process. The time constant τ determines how quickly $V_c(t)$ reaches the generator voltage V_s, and its impact is less evident in the latest stage of the charging curve. Thus, sampling data in different time intervals may affect the functional dependence approximation. To investigate *how data sampling affects the linear coefficient estimation*, we generate 10 data points in different temporal regions of the charging curve. We consider intervals of the form $[0, EOH]$, where $EOH \in (0, 5\tau]$ refers to the end-of-horizon of the measurements; since τ changes over time, we consider the maximum as a reference value to compute the EOH.

As shown in Figure 4.3, the linear model approximation is more accurate if the data points are sampled in an interval with $EOH \in [1.5\tau, 3\tau]$, where $V_c(t)$ approximately reaches respectively the 77% and 95% of V_s. With higher values of EOH, the sampled data points are closer to the regime value V_s, and the impact of τ is less relevant in the system state evolution. Thus, the learning model can achieve high prediction accuracy of $V_c(t)$ without correctly learning the functional dependence.

Non-Linear Dependence Here, we test the UDE framework under the assumption of a non-linear dependence between the observable and the β parameter of the epidemic model. The observable is a set of Non-Pharmaceutical Interventions (NPIs), which affects the virus spreading at each time step. To generate the epidemic data, we define the following time series representing the variation at time t of the inherent infection rate of the disease, $\hat{\beta}$, under the effect of two different NPIs per time instance:

$$\beta(t, \mathbf{x}^t, \mathbf{e}, \hat{\beta}) = \hat{\beta} \cdot e_1^{x_1^t} \cdot e_2^{x_2^t} \tag{4.11}$$

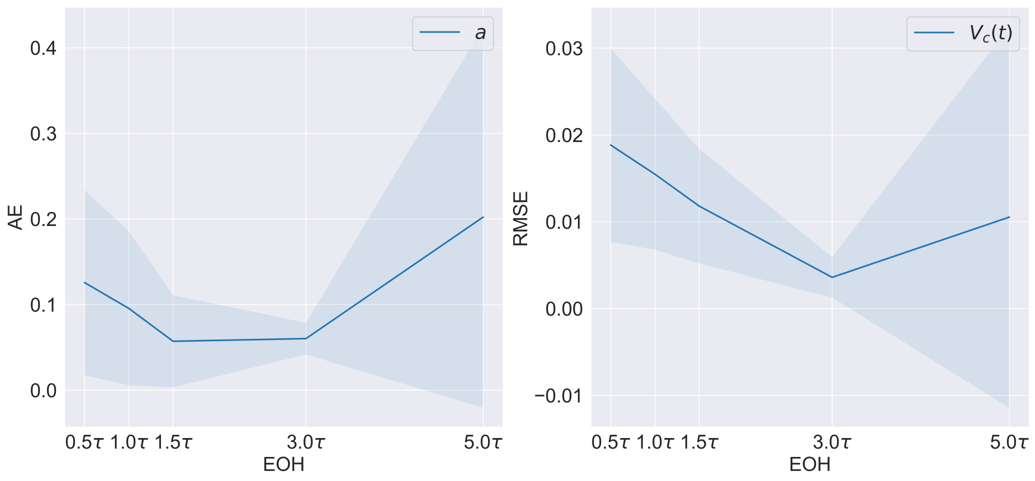


Figure 4.3: Linear coefficients and predictions error and a function of the EOH.

where $\mathbf{x}^t \in \{0,1\}^2$ is the binary vector indicating whether the corresponding NPIs are active at time t. The vector $\mathbf{e} \in [0,1]^2$ represents the effects of the two NPIs in reducing the infection rate. We compute 100 different time-series for β by assuming that the vector of NPIs, $\mathbf{x}^t$, randomly changes each week. For each of the resulting time series, we generate 20 data points equally spaced from day 0 to day 140 of the outbreak evolution. The generation process relies on a highly accurate Euler integration with 10.000 iterations per time step and uses the same initial condition and γ value described in the *Training Procedure* experiment (4.1.1). To approximate the non-linear dependence in 4.11, we rely on a DNN which forecasts the value of β based on $\mathbf{x}^t$ and the state of the system at time $t-1$. Thus, the resulting universal approximator of the UDE framework is a black-box model able to capture complex functional dependencies, but lacking interpretability.

The experimental results show that the UDE framework can estimate the dynamic system state with high accuracy (the RMSE of the state prediction is 0.037±0.013); however, the model is unable to provide an accurate estimation of the β time-series, which RMSE is equal to 0.37 ± 0.17. Similarly to the RC-circuit, we investigate the effect of the data sampling frequency on the parameter approximation accuracy of the UDE. We consider 4 different time horizons, namely $5, 10, 15$, and 20 weeks of the outbreak evolution, in which we sample 20 equally spaced data points. We train the model on the resulting data, and we compute the reconstruction error (RMSE) on the complete epidemic curve of 20 weeks. We report both the parameter approximation error and the curve reconstruction error in Figure 4.4.

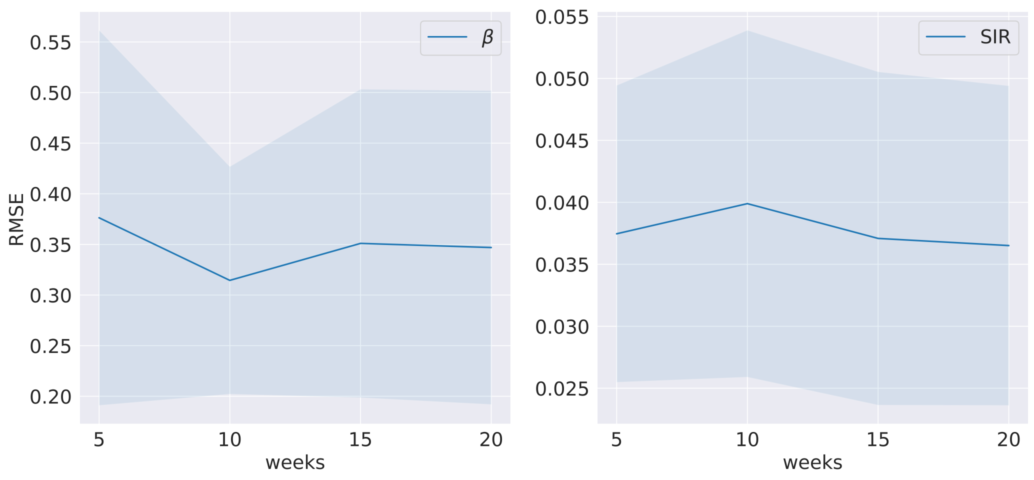


Figure 4.4: Non-Linear dependence: β and prediction errors with different sampling frequencies.

Conversely to RC-circuit, the sampling process does not seem to have a significant impact on the model accuracy. The reason for this result may be found in the complexity of the function to be approximated, and in the impact of β parameter on the epidemic curve. In the RC-circuit, the system state evolution is an exponential function of the unknown parameter τ, and we can design the collection process to cover the temporal interval where the impact of τ is more relevant. In the SIR model, we do not have a closed-form of the epidemic curve, and thus it is harder to select the most relevant temporal horizon.

4.1.2 Parameter Estimation with Universal Differential Equations

In this section, we introduce an extension of the FULL-BATCH method. Our primary focus in this set of experiments is to estimate the parameters of a compartmental model used to describe the epidemic. In a predictive context, our objective is to forecast the trajectory of the epidemic. In this context, we seek to unveil the underlying dynamics of disease spread within the framework of the dynamic system representing the outbreak.
For the sake of simplicity, we confine our analysis to the SIR (Susceptible-Infectious-Recovered) model, however, it can be easily extended to other compartmental models. In its general formulation, a SIR model necessitates the specification of the values for two critical parameters, namely β and γ. Both of these parameters are time-dependent, and subject to variations con-

tingent on changing contextual circumstances. To streamline our analysis, we make the simplifying assumption that γ tends to remain relatively constant during an epidemic. This is based on the notion that the average recovery period is primarily influenced by clinical and treatment factors, which tend to stabilize once medical practices are standardized. For example, multiple studies have indicated that the average recovery time for COVID-19 falls within the range of 14 to 16 days (Barman et al., 2020; Bhapkar et al., 2020). In contrast, we acknowledge that β exhibits significant variability, influenced by dynamic factors such as environmental conditions, containment measures, and ultimately, human behavior. In light of these assumptions, we assign a fixed value to γ (specifically, $\frac{1}{15}$), while representing β as a time-dependent variable denoted as $\beta(t)$. To estimate the function $\beta(t)$, we design a custom neural network model based on the method. This model comprises several layers equal to the time steps (e.g., days) covered by the available data, denoted as T. Each layer corresponds to the state of the SIR model at time i, and a sequence of T weights encapsulates our current understanding of the model's parameterization, specifically $\beta(t)$. The architecture of the initial model used to fit the parameterization of the SIR curve is depicted in Figure 4.5.

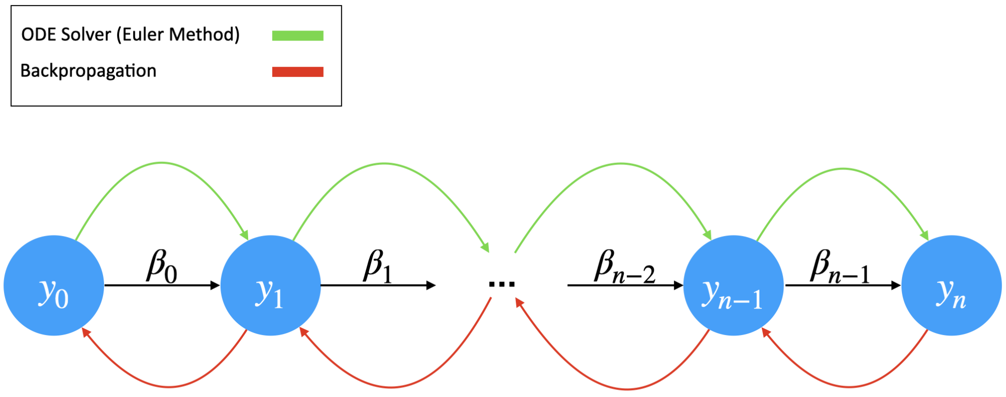


Figure 4.5: Architecture of the initial model employed to fit the parameterization of the SIR curve

The forward phase of this model involves solving the SIR dynamic system using the parameterization belief and an initial state, denoted as $y_0 = (S_0, I_0, R_0)$. This allows us to compute a cost function that guides the optimization process, leading to updates in the network weights, which, in this case, represent our belief about the β series. The loss function used in this optimization process is defined as follows:

$$Loss(y, \hat{y}) = \frac{1}{T}\sum_{t=0}^{T}(y_t - \hat{y}_t)^2 + \lambda\Omega_\beta$$

Here, $y_i = (S_i, I_i, R_i)$ represents our observed data, $\hat{y}_i$ signifies the output of the forward step, Ω_β is a term that enforces smoothness in the β series, and λ serves as the associated weighting coefficient. The regularization term, Ω_β, is instrumental in imposing a soft constraint on β to prevent abrupt changes:

$$\Omega_\beta = \frac{1}{T-1}\sum_{t=1}^{T} max(0, \beta_t - \beta_{t-1})$$

Significant deviations in the β series are penalized by the regularization term within the loss function. The weight of the regularization term, denoted as λ, is optimized during the training process using the Lagrangian dual method based on sub-gradient descent [58], presented in Section 2.2.3. The inclusion of this penalty term aligns with empirical observations suggesting that abrupt changes in the infection rate are rare.

Experimental Results

We evaluated the model using synthetic datasets, created to mimic various patterns in the β historical series. Starting with a set of β_i values for $i = 1, 100$, along with initial conditions, we generated the corresponding SIR curves. Our objective was to establish a realistic β series by:

- Constraining the values to fall within the range of $[0.11, 0.15]$. This decision was informed by observed trends in the basic reproduction number, R_0, throughout the COVID-19 pandemic [103].
- Producing six distinct sets of curves, each derived from a unique distribution, to encapsulate a variety of potential scenarios.

Table 4.3 encapsulates the results of our experiments, quantified using the *mean squared error* (MSE) and *mean absolute percentage error* (MAPE).

We repeated the fitting procedure with diverse initial assumptions regarding the β series, calculating both metrics for each of the six generated series. The results, encompassing means, and standard deviations, are detailed in Table 4.3. The experimental outcomes affirm a notable degree of accuracy across all the curves, as illustrated in Figure 4.6, and corroborated by the

height β Generation Distribution	MSE	MAPE
Gaussian	0.00084 ± 0.0002	0.32 ± 0.049
Exponential Gaussian	0.00012 ± 0.00012	0.11 ± 0.052
Inverse Exponential Gaussian	$0.00014 \pm 4.56 \cdot 10^5$	0.11 ± 0.025
Sigmoid Gaussian	$0.00010 \pm 6.51 \cdot 10^5$	0.094 ± 0.022
Exponential Sigmoid Gaussian	0.00010 ± 0.00013	0.078 ± 0.051
Inverse Exponential Sigmoid Gaussian	$9.77 \cdot 10^5 \pm 4.44 \cdot 10^5$	0.10 ± 0.025

Table 4.3: Evaluation Metrics Fitting β: MSE and MAPE

data in Table 4.3. Notwithstanding, it is worth noting that the model exhibited challenges in approximating the curves generated from the Gaussian mixture, primarily struggling to accurately trace all the peaks and troughs. Despite this, the model succeeds in capturing the overarching trend of the series. The observed difficulty in curve fitting for this experimental case could be attributed to the data's high variance, which could be hard to capture since penalized by the regularization term.

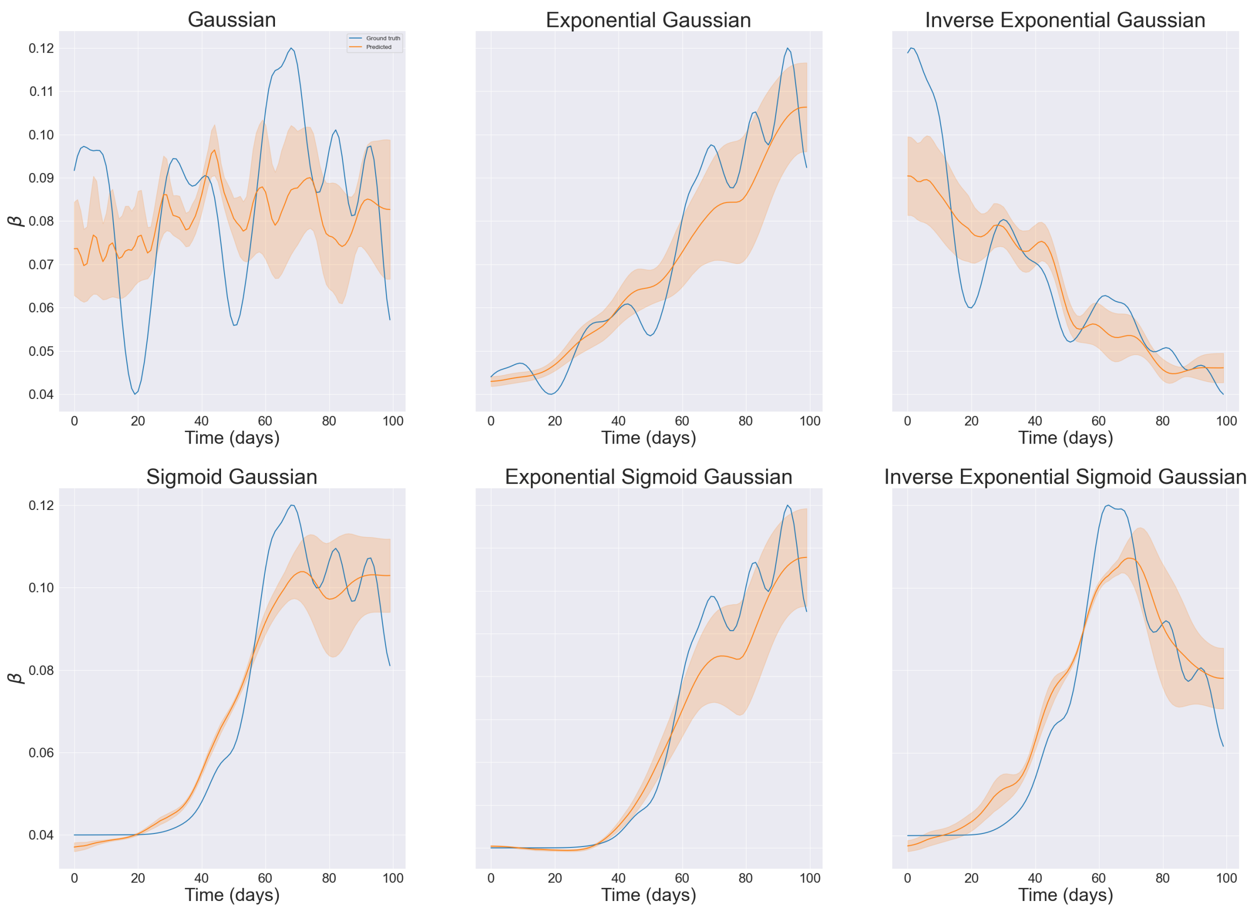


Figure 4.6: Fit β

As highlighted by [8], the approximated β values can serve as supervisory signals for training an additional machine learning model. This methodology can be seamlessly integrated into an end-to-end framework, utilizing the MINI-BATCH approach to enhance the overall predictive performance and efficiency, this solution will be further explored in the next section.

4.2 Renewal Equations

This section introduces a method that integrates classical renewal equations with DNNs, aiming to enhance the predictive accuracy of infectious disease progression over time. We utilize Renewal Equations, widely deployed in the field of epidemic forecasting (Section 2.3.1), as a structural basis to construct a neural model. This architecture is designed to predict the trajectory of infection cases, utilizing a finite historical series that captures the development of the outbreak. We start considering the discrete formulation of renewal equations, expressed as follows:

$$y_t = R(t-1) \sum_{s=t-l}^{t-1} y_{t-s}\, \omega(s) \tag{4.12}$$

Here, y_i signifies the daily count of infections, while ω represents the inter-arrival distribution or the generation time distribution, characterized as a Probability Mass Function (PMF). Furthermore, we operate under the assumption of knowing the new infections within a finite time window, denoted as $[t-l, t-1]$.
In this context, the parameter l, referred to as our *lookback window*, indicates the extent of historical information assumed to be available for making predictions at a given moment. This choice has two main reasons: on one hand, information from an extended period in the past may not be as pertinent for forecasting at a given point; on the other hand, it reduces computational complexity by processing a smaller data sample.
Hence, based on these assumptions, our learning objective is to approximate the function:

$$f_\theta(y_{t-l}, y_{t-l+1}, ..., y_{t-1}) = y_t \tag{4.13}$$

The neural architecture adopted here rests upon two key approximations:

Reproductive Factor. The R_t factor, a measure of the infection's transmission rate at time t, is estimated using a neural model that receives the lookback window as its input. This model may incorporate additional inputs such as temperature, mobility data, and interventions.

Generation Time Distribution. The generation time distribution, ω, modeled as PMF, is approximated with a mechanism involving an array of trainable probabilities that can be accessed using the corresponding time value. The vector, denoted as $\mathbf{w}$, has a length of l, reflecting the historical window considered and must satisfy the following properties:

1. All probabilities must be non-negative: $\forall w_i \in \mathbf{w}, w_i \geq 0$
2. The probabilities of all possible values of the random variable should sum up to 1: $\sum_i w_i = 1$

The first property can be enforced by considering the exponential values of the weights, e^{w_i}, ensuring that the probabilities are strictly greater than zero. To meet the second condition, we normalize the weights using the softmax function:

$$\sigma(w_i) = \frac{e^{w_i}}{\sum_j e^{w_j}} \tag{4.14}$$

Consequently, we can re-write the renewal formulation as:

$$y_{t+1} = R_\theta(t-1) \sum_{s=l}^{t-1} y_{t-s}\ \sigma(\mathbf{w}) \tag{4.15}$$

where θ denotes the parameters of the network approximating the R_t factor.

Experimental Results

The historical series considered in the experimental phase is relative to daily or weekly variations in the number of infected within a heterogenous population during an outbreak. The time series representing the epidemic is segmented based on a fixed *lookback window*. This results in the creation of training instances in the form of (X_t, y_t), where:

$$X_t = (y_{t-l}, y_{t-l+1}, ..., y_{t-1}) \tag{4.16}$$

, where y_i represents the daily or weekly count of new cases, while l is the fixed value assigned as the lookback period. The segmentation allows breaking the temporal dependence in the dataset, providing the means to have a more flexible learning task.
The primary objective is to predict the number of new infected cases at the next time step, given a small sample of data representing the past epidemic trend within a specified period. Assuming to have successfully approximated f_θ, we can provide a forecast over a longer time range feeding back the previous predictions into the model:

$$f_\theta(X_{t+i}) = \begin{cases} f_\theta(y_{t-l+i}, ..., y_{t-1}, f_\theta(X_t), ..., f_\theta(X_{i+i-1})), & \text{if } i \leq l \\ f_\theta(f_\theta(X_t - l + i), ..., f_\theta(X_{i+i-1})), & \text{otherwise} \end{cases} \tag{4.17}$$

This formulation reveals the *ill-conditioned nature of the predictive task*, as any errors in the predictions can propagate and amplify over time.

To evaluate the performance of our model, we conducted a comprehensive analysis, comparing its accuracy with a variety of baseline methodologies, namely:

- **AutoRegressive Integrated Moving Average (ARIMA)**: This method is a popular choice for forecasting time series data, known for its versatility and effectiveness across different types of series, including those related to epidemics [119, 124].
- **Bidirectional-LSTM**: This neural network structure is designed to capture complex temporal dependencies in time series data by considering both past and future information. We have previously discussed its applications in epidemiology in Section 2.3.2.
- **Universal Differential Equations and SEIR Model**: In this instance, we have extended the capabilities of the MINI-BATCH method to accommodate the SEIR epidemiological model. This necessitated a modification of the architectural framework to incorporate additional degrees of freedom. Specifically, the model is now tasked with not only inferring the parameters of the compartmental model but also simultaneously predicting the states of the compartments themselves:

  $$y_t = U_\theta(y_{t-1}, t-1)) \tag{4.18}$$

  where, $y_t = (S_t, E_t, I_t, R_t, \beta_t, \epsilon_t, \gamma_t)$. This modification invariably results in a higher complexity of the learning task. The rationale behind this decision is intrinsically tied to the characteristics of the data in consideration. Typically, the MINI-BATCH method exhibits optimal performance when all compartments are available for supervision during the training phase. However, this condition is often unmet in real-world data scenarios.

The evaluation includes a set of metrics that assess the disparity between the approximation generated by the trained model and the ground truth. The quantities used to measure the models' performance are:

- **Mean Absolute Percentage Error (MAPE)**: This metric offers a computation of the average magnitude of error between the predicted number of infected cases and the actual figures, subsequently expressing this discrepancy as a percentage. This provides a normalized scale for error measurement, facilitating easier interpretation and comparison across different scenarios.

- **Pearson Correlation Coefficient** (ρ): Serving as a statistical measure of linear correlation, this metric evaluates the degree to which two data sets are linearly related, providing valuable insights into the similarity of the temporal patterns they exhibit. Its efficacy in time series forecasting, particularly in terms of aligning and comparing trendlines, is well-established.

Except for the ARIMA model, the resilience and stability of each method under consideration are tested using different random initializations. This allows us to measure and assess the robustness of the models. Additionally, the model is tested using increasing amounts of training data, mimicking a real emergency scenario in which data become progressively available as the outbreak unfolds. This approach allows us to estimate the model's performance at different phases of the outbreak.

Metapopulation simulation of COVID-19 in Frances. This dataset represents a simulated outbreak in 12 regions of France spanning over 265 days. Each historical series represents the *daily* new infected. The curves were generated using a discrete stochastic transmission model with a metapopulation structure at the regional level. This model, developed by INSERM [136], incorporates various critical factors such as demography, mobility, seasonality, the frequency of the Alpha variant, and vaccination. The transmission dynamics are characterized by a compartmental scheme specifically designed for COVID-19, which categorizes individuals into susceptible, exposed, infectious, hospitalized, and recovered states. These compartmental models were previously employed to address the COVID-19 pandemic in France in 2020 [43, 46, 44, 125, 45], for tasks such as assessing the impact of lockdown [43], the night curfew [46], and the reopening of schools [44], estimating the underdetection of cases [125], and anticipating the impact of the Alpha variant in France [45]. For the experiments on this dataset, we fixed the *lookback window* to 7 days, while we evaluated the performance of the model on a prediction of 21 days.

Season Flu in Italy. The dataset contains information regarding seasonal flu in Italy at a regional level. The data were collected by the Italian National Institute of Health (ISS) and are publicly available at the provided link[1]. The dataset consists of *weekly* new cases of seasonal flu in the 21 Italian regions, covering the period from 2011 to 2023. However, we decided to restrict our analysis to data previous to the COVID-19 pandemic, thus before 2020. For the experiments of this dataset, we fixed the *lookback window* to 2 weeks, while we evaluated the performance of the model on a prediction of 8 weeks.

Discussion

The results presented in Figure 4.7, and 4.8 show the evaluated metrics for predictions at different time horizons. At Table 4.4, we summarize the evaluation metrics for the prediction at 7 days for the French dataset and 4 weeks for the Italian one.
The deep renewal model is the next best option for the French scenario, followed by the Bi-LSTM (Figure 4.7). Indeed, both models show good performance across different splits, and just in a few cases are outperformed by other models, such as the ARIMA, but still marginally.
Concerning Italy's national flu data (Figure 4.8), again the DL model proved to be among the best performing, followed closely by the ARIMA model, which has comparable performance. In both scenarios, in presence of a lower amount of data (namely, the 40% split), the renewal-informed learning process provides good performance validating the use of an informed learning process. Moreover, both the Bi-LSTM and deep renewal equations achieve good results in terms of correlation, underscoring their ability to approximate the overall trend more effectively. The renewal-based model holds a marginal advantage in this regard. The note of demerit goes to the UODE-based method, which is the worst performing across all the experimental instances. This is most likely due to the increased complexity of the model, which makes it much harder to approximate the epidemic trend.
Lastly, we can observe that there is a discernible trend that indicates deteriorating performance as we expand the time horizon of the prediction, which validates the ill-conditioned hypothesis made earlier.

Overall, the deep renewal model demonstrates performance comparable to other DL-based methodologies and in a few cases outperforms the baselines. Moreover, we should also underline that this method offers two potential advantages that should be further investigated.

- It offers higher levels of accuracy in the presence of fewer training

[1]https://github.com/fbranda/influnet

instances, in particular, if compared to other DL methods (e.g. Bi-LSTM).

- The Deep Renewal Equation model is inherently interpretable. The quantities estimated via DNNs possess meaningful interpretations in terms of virus spread mechanics. Notably, both the R_t factor and the Generation Time Distribution are significant in evaluating the contagiousness of the disease, providing clear and actionable insights.

In light of this potential, the Deep Renewal Equation model merits further investigation to fully capitalize on its potential benefits.

Method	Split					
	France					
	40%		60%		70%	
	MAPE (%)	ρ	MAPE (%)	ρ	MAPE (%)	ρ
ARIMA	5.15 ± 0.83	0.99 ± 0.00031	0.37 ± 0.001	0.99 ± 0.00063	4.74 ± 0.75	0.10 ± 0.22
Bi-LSTM	38.58 ± 7.42	0.096 ± 0.23	10.03 ± 1.003	0.17 ± 0.26	14.15 ± 1.23	-0.10 ± 0.24
UODE	83.27 ± 18.29	0.12 ± 0.18	36.38 ± 24.14	-0.016 ± 0.21	57.44 ± 47.15	-0.18 ± 0.19
DRE	52.23 ± 2.08	0.71 ± 0.10	6.21 ± 1.34	0.39 ± 0.22	11.78 ± 1.24	-0.11 ± 0.11
Bi-LSTM + 0-shot	28.96 ± 10.98	-0.20 ± 0.32	50.39 ± 10.98	0.094 ± 0.25	21.35 ± 9.56	0.20 ± 0.16
UODE + 0-shot	33.79 ± 15.22	0.11 ± 0.23	30.45 ± 3.84	0.38 ± 0.15	17.50 ± 2.16	0.42 ± 0.12
DRE + 0-shot	26.81 ± 3.6	0.82 ± 0.22	35.73 ± 5.03	0.95 ± 0.057	10.76 ± 5.19	0.65 ± 0.15
Bi-LSTM + K-shot	66.88 ± 12.83	0.41 ± 0.16	186.065 ± 48.63	0.34 ± 0.13	55.77 ± 14.86	0.41 ± 0.13
UODE + K-shot	273.47 ± 52.21	-0.53 ± 0.12	92.75 ± 43.99	0.043 ± .064	151.77 ± 46.04	0.28 ± 0.17
DRE + K-shot	49.99 ± 7.24	0.71 ± 0.14	253.0083 ± 52.64	0.13 ± 0.23	74.32 ± 23.62	0.37 ± 0.25
	40%		60%		70%	
	Italy					
	MAPE (%)	ρ	MAPE (%)	ρ	MAPE (%)	ρ
ARIMA	396.72 ± 158.40	0.34 ± 0.12	25.47 ± 3.06	0.16 ± 0.14	35.68 ± 3.50	0.10 ± 0.20
Bi-LSTM	118.79 ± 51.96	0.15 ± 0.24	25.40 ± 4.0063	0.15 ± 0.13	63.46 ± 3.94	0.75 ± 0.16
UODE	130.12 ± 49.022	-0.18 ± 0.13	233.28 ± 46.38	-0.32 ± 0.15	251.99 ± 87.47	0.024 ± 0.21
DRE	24.38 ± 3.0063	0.76 ± 0.11	44.58 ± 5.68	0.25 ± 0.12	80.61 ± 21.55	0.95 ± 0.026
Bi-LSTM + 0-shot	50.85 ± 10.87	0.17 ± 0.10	53.83 ± 15.17	10.72 ± 0.12	60.44 ± 11.26	0.57 ± 0.27
UODE + 0-shot	303.29 ± 51.41	-0.50 ± 0.05	156.56 ± 53.27	-0.24 ± .33	201.14 ± 32.22	0.10 ± 0.35
DRE + 0-shot	55.96 ± 8.27	-0.060 ± 0.21	120.27 ± 31.64	0.73 ± 0.13	53.28 ± 12.47	0.58 ± 0.24
Bi-LSTM + K-shot	66.88 ± 12.83	0.41 ± 0.16	186.065 ± 48.63	0.34 ± 0.13	55.77 ± 14.86	0.41 ± 0.13
UODE + K-shot	273.47 ± 52.21	-0.53 ± 0.12	92.75 ± 43.99	0.043 ± .064	151.77 ± 46.04	0.28 ± 0.17
DRE + K-shot	49.99 ± 7.24	0.71 ± 0.14	253.0083 ± 52.64	0.13 ± 0.23	74.32 ± 23.62	0.37 ± 0.25

Table 4.4: Evaluation metrics for prediction on French data at 7 days and on Italian data at 4 weeks: MAPE and ρ; Deep Renewal Equations (DRE); Bidirectional LSTM (Bi-LSTM); Universal Ordinary Differential Equation (UODE)

4.3 Transfer Learning

A relevant declination of ML is Meta Learning, which translates to the formula "learn to learn" [81, 144]. What this framework proposes is to learn a generalization among multiple related tasks. For instance, we might want to

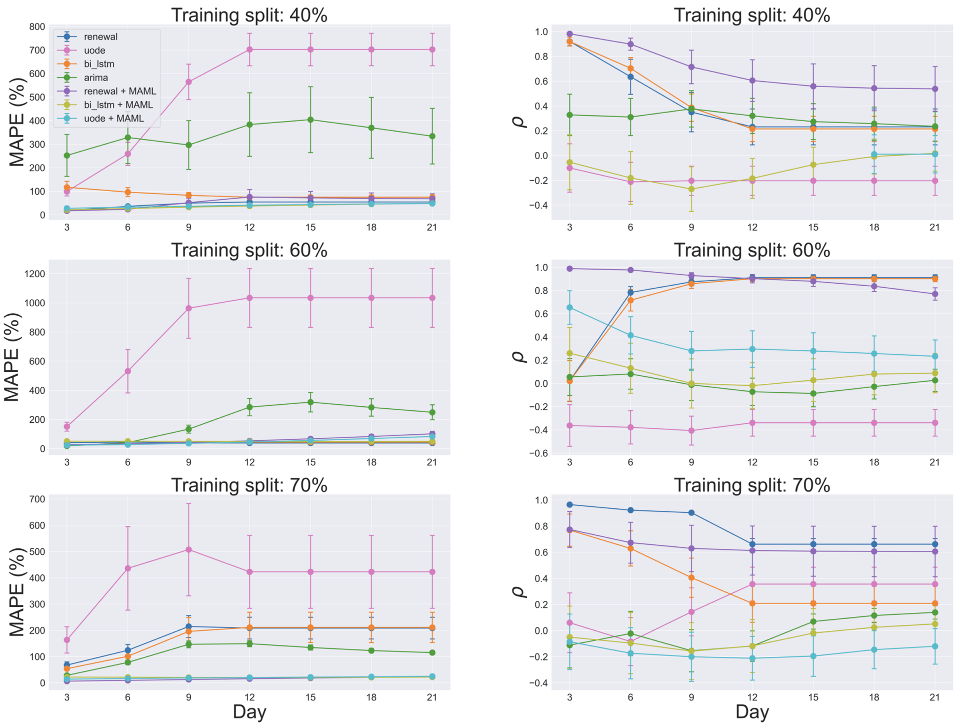


Figure 4.7: On the left: MAPE. On the right: Pearson Correlation - France

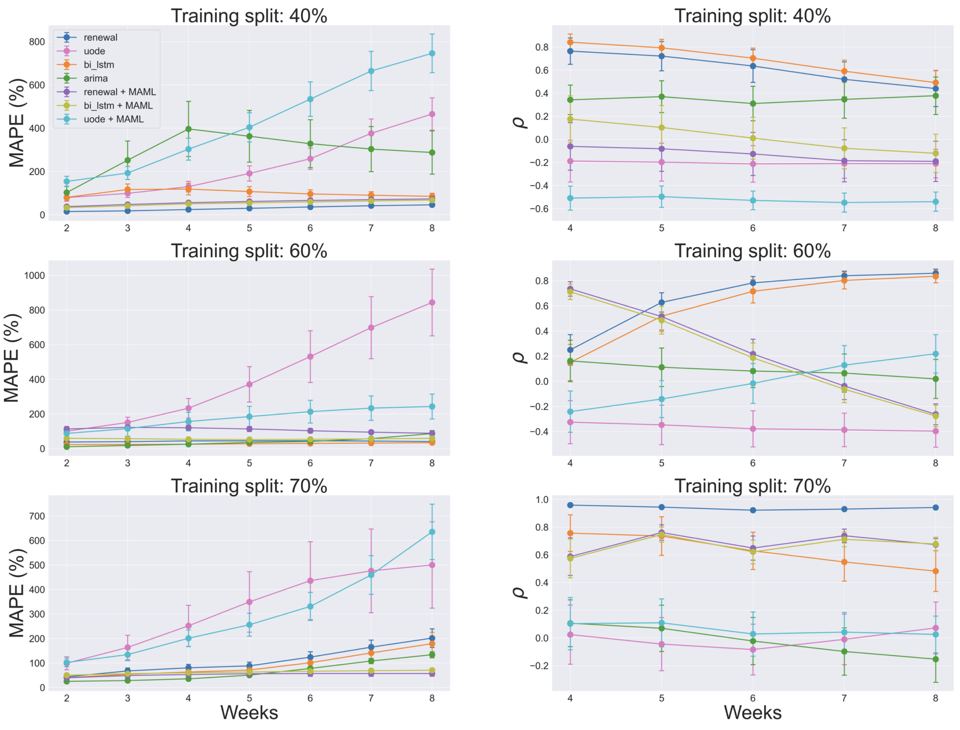


Figure 4.8: On the left: MAPE. On the right: Pearson Correlation - Italy

approximate the dynamics of multiple airborne diseases to forecast their evolution in time - the idea would be to exploit the data coming from different diseases to obtain better models. Meta-Learning comprehends another class of approaches called Transfer Learning. These methods provide the means to *transfer* knowledge from one ML model to the other, possibly across different tasks; this means that the approximation of a task provided by one model could be the starting point for training or refining a different model on a similar domain [111, 133]. An example might be transferring the predictive capability from one disease to another. An extreme declination of this technique is Zero-shot Learning, where the objective is to use a model trained on a task to predict another one (for which we have no data), e.g., we might have approximated the dynamics of COVID-19 using a DNN and use such model to predict the trend of seasonal flu. This approach can be generalized to Few-shot Learning, which instead infers knowledge regarding a phenomenon for which we have a small data set [122]. Both these methods can tackle core issues of epidemiology, where often data samples are scarce or, in some cases, inexistent. In the scope of our work, we will mainly focus on the approach adopted in [122], with particular attention to the performance of the method in 0-shot and Few-shot learning scenarios.

4.3.1 Model-Agnostic Meta Learning

The transfer learning method applied in [122] is Model-Agnostoc Meta-Learning (MAML) [57]. MAML aims to enable fast adaptation of a model to new tasks with a small amount of training data. The key idea is that we can learn the initialization of model parameters that facilitate the adaptation to a novel setting. To do so, we are required to provide a set of learning tasks that are reasonably linked to our target task. For instance, data regarding weather are hardly useful in providing transfer knowledge for epidemics forecasting. The intuition is that by training the model on a diverse set of tasks and encouraging it to learn an initialization that is good for adaptation, it becomes more gradient-friendly for a wide range of tasks. This means the model should be close to a good solution for many tasks in its parameter space, making it easier to adapt to specific tasks with minimal fine-tuning.

The fundamental concept behind Model-Agnostic Meta-Learning (MAML) can be concisely encapsulated in a few steps, as illustrated in Algorithm 2. MAML commences with the establishment of a *base model*, typically initialized with randomly assigned weights. In the subsequent meta-training stage, this base model undergoes exposure to a diverse array of tasks, each defined by a small dataset comprising input-output pairs. The primary objective of this meta-training phase is to optimize the base model such that it be-

Algorithm 2 Model-Agnostic Meta-Learning (MAML)

```
1: procedure MAML($\alpha$, $\beta$, epochs)
2:     Input: $\alpha$, inner learning rate; $\beta$, outer learning rate
3:     Initialize model parameters: $\theta$ (Random initialization)
4:     for each epoch do
5:         Sample a batch of meta-tasks: $\mathcal{T} = \{\tau_1, \tau_2, \dots, \tau_k\}$
6:         for each task $\tau_i \in \mathcal{T}$ do
7:             Split the task into support ($D_{\text{train}}$) and query ($D_{\text{test}}$) sets.
8:             for each gradient step do
9:                 $\theta' \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, D_{\text{train}})$
10:            end for
11:            Evaluate the model on query set: $L(\theta', D_{\text{test}})$
12:        end for
13:        $\theta \leftarrow \theta - \beta \nabla_\theta \frac{1}{k} \sum_{i=1}^{k} L(\theta', D_{\text{test}})$
14:    end for
15: end procedure
```

comes a proficient learner for one of the tasks, achievable through a limited number of gradient-descent steps. Following this, each task-specific model undergoes evaluation using an unseen data sample drawn from the corresponding transfer task. This meta-testing phase facilitates the computation of a new gradient, which is subsequently utilized to refine and update the initial base model. This iterative process is perpetuated on the base model until a predefined stopping criterion is met.

Experimental Results

In this work, we investigated the effect of transfer learning between adjacent geographical regions. In particular, we will focus on regions within the same country, namely, France and Italy. The idea is to use part of the epidemic curves relative to some regions to approximate the trend of another one, which represents our target. Thus, for each training instance, we randomly selected 5 regions based on the country considered and rotated the target region among 5 other possible candidates - which are different from the one chosen as transfer knowledge. The sampled data are then further divided in D_{train} and D_{test}.

The general experimental setup is similar to the one presented in Section 4.2. The baselines, the preprocessing, the metrics, and the testing phase follow the same general approach.

At this point, we can perform two distinctions in the transfer learning process:

- **0-shot Learning**: The model is trained using the MAML algorithm and used to predict the target task. This means that the model is predicting without having any knowledge regarding the target region.
- **Few-shot Learning**: The models are trained using the MAML algorithm and then further trained on a few data samples regarding the target region. This second training phase requires fewer data and requires less training iteration, if compared to classic gradient-descent.

Discussion

The results presented in Figure 4.7, and 4.8 show the evaluated metrics for predictions at different time horizons for the 0-shot case, while in Figure 4.9 we provide a small example of a forecast for the flu season 2015-2016 in the region of Tuscany over 5 weeks. While, at Table 4.4, we summarize

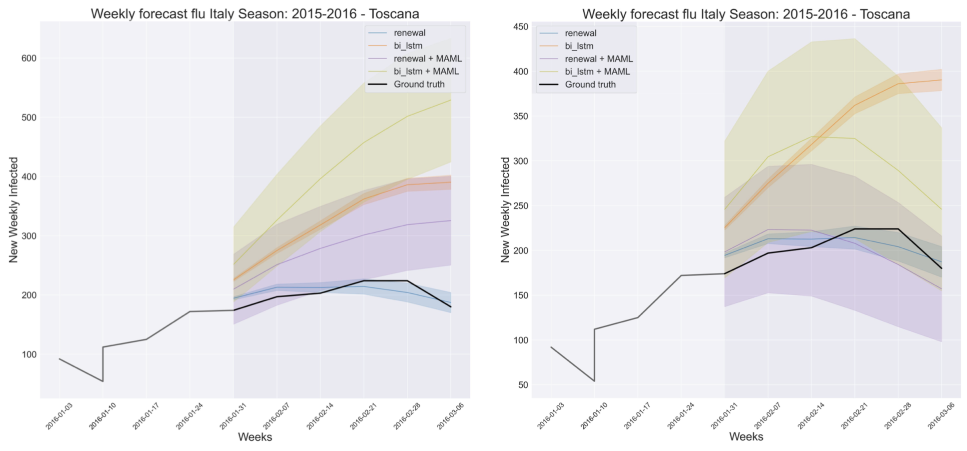


Figure 4.9: Comparison of Bi-LSTM and deep renewal equations, along with their 0-shot versions on the left, and Few-shot versions on the right, in forecasting the flu season for 2015-2016 over 5 weeks in the Tuscany region.

the evaluation metrics for the prediction at 7 days for the French dataset and 4 weeks for the Italian one. About the synthetic data about France, the implementation of the transfer learning algorithm has resulted in comparable performance to the one presented in the section before. However, we should underline that these outcomes represent the averaging of the 0-shot prediction's error among different regions. Thus information regarding the

outbreak in the target region itself is not explicitly provided at training time. Nevertheless, the application of transfer learning results in a reduction of the correlation values of the curves and increased variability (Figure 4.7, 4.7).
A separate consideration must be made for the UODE model, which showed a strong increase in performance when paired with the MAML algorithm, providing performance comparable to the other DL models for the French scenario.
Regarding K-shot learning (or, Few-shot learning), we can see how the performance does not improve steadily, except for the UODE model. Indeed, as shown in Figure 4.9, the retraining process results in milder alignment with the outbreak in our target regions.
The Italian case study provides similar results to the French scenario with slightly worse performance but in line with the general trend.

The findings outlined in this section provide a preliminary exploration into the possible implementation of transfer learning within the epidemic forecasting field. Employing these techniques in this domain holds significant promise for addressing many of the challenges inherent to data-driven approaches in predicting the progression of outbreaks. The experiments detailed here primarily concentrate on the transfer of knowledge between adjacent geographical regions. However, the scope of the application has the potential for considerable extensions:

- How the knowledge at a regional level transfer at the country level?
- Can the transfer process be also applied among different seasons of the same outbreak?
- Past outbreak could represent a powerful source of data for the transfer process. Investigating diseases with similar profiles as sources of knowledge for unknown illnesses could hold high potential to improve the field of epidemiology.

Chapter 5

Conclusion

represents an effort to devise Machine Learning-based strategies aimed at addressing challenges posed by the emergence of novel viruses, a phenomenon anticipated to escalate in the forthcoming decades. The methodologies discussed predominantly hinge on repurposing existing knowledge, particularly from classical epidemiology, to enhance the learning process. Two principal domains have been identified where ML can be employed as a tool for enhancement:

- **Data Analysis**: Machine Learning-based solutions can deepen our understanding of clinical records concerning infected individuals, uncovering new patterns and augmenting our comprehension of specific conditions associated with diseases. However, the application of these methodologies is frequently obstructed by issues related to the availability and quality of data. To mitigate this challenge, we have introduced two methodologies aimed at improving the quality and reliability of data analysis through tree-based models.

- **Predictive Epidemiology**: In this realm, we have explored two distinct methodologies. On the one hand, we utilized Universal Differential Equations in conjunction with compartmental models; while on the other hand, we introduced a new deep learning approach embedding renewal equations in the learning process. Moreover, we have extended these methods through the integration of transfer learning. This opens the door to the possibility of leveraging knowledge from other sources and enhancing the performance of these methods.

This work is an attempt to interject a trend in which AI will be an increasingly pervasive element of progress in our society. In this context, AI holds the potential to be the tool that equips us to confront future challenges, such as pandemics. Indeed, provides us with the necessary to avert another catastrophic event like the COVID-19 outbreak. With this perspective, we want to highlight how the scientific community points to an increased likelihood of the surge of nove zoonotic viruses. To this end, it becomes increasingly important to foster cross-disciplinary collaborations, delving into how traditional methodologies can be refined and advanced through the innovative lens of Machine Learning.

Bibliography

[1] B. Adhikari, X. Xu, N. Ramakrishnan, and B. A. Prakash. *EpiDeep: Exploiting Embeddings for Epidemic Forecasting*, page 577–586. Association for Computing Machinery, New York, NY, USA, 2019.

[2] Nurshad Ali. Elevated level of c-reactive protein may be an early marker to predict risk for severity of covid-19. *Journal of Medical Virology*, 92(11):2409–2411, 2020.

[3] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 136–145. JMLR. org, 2017.

[4] Martin Atzmueller and Eric Sternberg. Mixed-initiative feature engineering using knowledge graphs. In *Proceedings of the Knowledge Capture Conference*, pages 1–4, New York, NY, United States, 2017. Association for Computing Machinery.

[5] Babajide O Ayinde, Ehsan Hosseini-Asl, and Jacek M Zurada. Visualizing and understanding nonnegativity constrained sparse autoencoder in deep learning. In *International Conference on Artificial Intelligence and Soft Computing*, pages 3–14. Springer, 2016.

[6] Babajide O Ayinde and Jacek M Zurada. Deep learning of nonnegativity-constrained autoencoders for enhanced understanding of data. *arXiv preprint arXiv:1802.00003*, 2018.

[7] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[8] F. Baldo, M. Iannello, M. Lombarid, and M. Milano. Informed deep learning for epidemics forecasting. In *PAIS 2022: 11th Conference on Prestigious Applications of Artificial Intelligence, 25 July 2022, Vienna, Austria (co-located with IJCAI-ECAI 2022)*, volume 351, page 86. IOS Press, 2022.

[9] Federico Baldo, Lorenzo Dall'Olio, Mattia Ceccarelli, Riccardo Scheda, Michele Lombardi, Andrea Borghesi, Stefano Diciotti, and Michela Milano. Deep learning for virus-spreading forecasting: A brief survey. *arXiv preprint arXiv:2103.02346*, 2021.

[10] Federico Baldo, Allison Piovesan, Marijana Rakvin, Giuseppe Ramacieri, Chiara Locatelli, Silvia Lanfranchi, Sara Onnivello, Francesca Pulina, Maria Caracausi, Francesca Antonaros, Michele Lombardi, and Maria Chiara Pelleri. Machine learning based analysis for intellectual disability in down syndrome. *Heliyon*, 9(9):e19444, 2023.

[11] Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep cnns? In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 4266–4276. Curran Associates Inc., 2018.

[12] R. G. Bengis, F. A. Leighton, J. R. Fischer, M. Artois, T. Morner, and C. M. Tate. The role of wildlife in emerging and re-emerging zoonoses. *Revue scientifique et technique-office international des epizooties*, 23(2):497–512, 2004.

[13] Daniel Berrar, Philippe Lopes, and Werner Dubitzky. Incorporating domain knowledge in machine learning for soccer outcome prediction. *Machine Learning*, 108(1):97–126, 2019.

[14] Tarek R. Besold, Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kuehnberger, Luis C. Lamb, Daniel Lowd, Priscila Machado Vieira Lima, Leo de Penning, Gadi Pinkas, Hoifung Poon, and Gerson Zaverucha. Neural-symbolic learning and reasoning: A survey and interpretation. 2017.

[15] Tom Beucler, Michael Pritchard, Stephan Rasp, Jordan Ott, Pierre Baldi, and Pierre Gentine. Enforcing analytic constraints in neural networks emulating physical systems. *Physical Review Letters*, 126(9):098302, 2021.

[16] Esben Jannik Bjerrum. Smiles enumeration as data augmentation for neural network modeling of molecules. *arXiv preprint arXiv:1703.07076*, 2017.

[17] T.D. Bjornsson. Use of serum creatinine concentrations to determine renal function. *Clin Pharmacokinet*, 4:200–222, 1979.

[18] Josh C. Bongard and Hod Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proc. Natl. Acad. Sci. USA*, 104(24):9943–9948, 2007.

[19] Andrea Borghesi, Federico Baldo, Michele Lombardi, and Michela Milano. Injective domain knowledge in neural networks for transprecision computing. In *Machine Learning, Optimization, and Data Science: 6th International Conference, LOD 2020, Siena, Italy, July 19–23, 2020, Revised Selected Papers, Part I 6*, pages 587–600. Springer, 2020.

[20] Andrea Borghesi, Federico Baldo, and Michela Milano. Improving deep learning models via constraint-based domain knowledge: a brief survey. *arXiv preprint arXiv:2005.10691*, 2020.

[21] Gert-Jan Both, Subham Choudhury, Pierre Sens, and Remy Kusters. Deepmod: Deep learning for model discovery in noisy data. *Journal of Computational Physics*, 428:109985, 2021.

[22] William Bradley and Fani Boukouvala. Two-stage approach to parameter estimation of differential equations using neural odes. *Industrial & Engineering Chemistry Research*, 60(45):16330–16344, 2021.

[23] Robin Burke. Knowledge-based recommender systems. *Encyclopedia of library and information systems*, 69(Supplement 32):175–186, 2000.

[24] T. Chakraborty, S. Chattopadhyay, and I. Ghosh. Forecasting dengue epidemics using a hybrid methodology. *Physica A: Statistical Mechanics and its Applications*, 527:121266, 2019.

[25] Kathleen Champion, Peng Zheng, Aleksandr Y Aravkin, Steven L Brunton, and J Nathan Kutz. A unified sparse optimization framework to learn parsimonious physics-informed models from data. *IEEE Access*, 8:169259–169271, 2020.

[26] Kathleen P. Champion, Peng Zheng, Aleksandr Y. Aravkin, Steven L. Brunton, and J. Nathan Kutz. A unified sparse optimization framework to learn parsimonious physics-informed models from data. *IEEE Access*, 8:169259–169271, 2020.

[27] David Champredon, Jonathan Dushoff, and David J. D. Earn. Equivalence of the erlang-distributed seir epidemic model and the renewal equation. *SIAM Journal on Applied Mathematics*, 78(6):3258–3278, 2018.

[28] Di Chen, Yiwei Bai, Wenting Zhao, Sebastian Ament, John M Gregoire, and Carla P Gomes. Deep reasoning networks: Thinking fast and slow. *arXiv preprint arXiv:1906.00855*, 2019.

[29] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[30] Zhao Chen, Yang Liu, and Hao Sun. Physics-informed learning of governing equations from scarce data. *Nature communications*, 12(1):1–13, 2021.

[31] Andrew Cotter, Maya Gupta, Heinrich Jiang, Erez Louidor, James Muller, Tamann Narayan, Serena Wang, and Tao Zhu. Shape constraints for set functions. In *International Conference on Machine Learning*, pages 1388–1396, 2019.

[32] Andrew Cotter, Maya Gupta, Heinrich Jiang, Nathan Srebro, Karthik Sridharan, Serena Wang, Blake Woodworth, and Seungil You. Training fairness-constrained classifiers to generalize, 2018.

[33] Andrew Cotter, Heinrich Jiang, Maya R Gupta, Serena Wang, Taman Narayan, Seungil You, and Karthik Sridharan. Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals. *Journal of Machine Learning Research*, 20(172):1–59, 2019.

[34] Andrew Cotter, Heinrich Jiang, and Karthik Sridharan. Two-player games for efficient non-convex constrained optimization. In *Algorithmic Learning Theory*, pages 300–332. PMLR, 2019.

[35] James P. Crutchfield and Bruce S. McNamara. Equations of motion from a data series. *Complex Syst.*, 1(3), 1987.

[36] Artur d'Avila Garcez, Marco Gori, Luis C. Lamb, Luciano Serafini, Michael Spranger, and Son N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. 2019.

[37] Luc De Raedt, Sebastijan Dumančić, Robin Manhaeve, and Giuseppe Marra. From statistical relational to neuro-symbolic artificial intelligence. *arXiv preprint arXiv:2003.08316*, 2020.

[38] Brian M de Silva, David M Higdon, Steven L Brunton, and J Nathan Kutz. Discovery of physics from data: universal laws and discrepancies. *Frontiers in artificial intelligence*, 3:25, 2020.

[39] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3844–3852, 2016.

[40] Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Lifted rule injection for relation embeddings. *ArXiv*, abs/1606.08359, 2016.

[41] Stephan Dempe and Susanne Franke. On the solution of convex bilevel optimization problems. *Computational Optimization and Applications*, 63(3):685–703, 2016.

[42] S. Deng, S. Wang, H. Rangwala, L. Wang, and Y. Ning. *Cola-GNN: Cross-Location Attention Based Graph Neural Networks for Long-Term ILI Prediction*, page 245–254. Association for Computing Machinery, New York, NY, USA, 2020.

[43] Laura Di Domenico, Giulia Pullano, Chiara E Sabbatini, Pierre-Yves Boëlle, and Vittoria Colizza. Impact of lockdown on covid-19 epidemic in île-de-france and possible exit strategies. *BMC medicine*, 18(1):1–13, 2020.

[44] Laura Di Domenico, Giulia Pullano, Chiara E Sabbatini, Pierre-Yves Boëlle, and Vittoria Colizza. Modelling safe protocols for reopening schools during the covid-19 pandemic in france. *Nature communications*, 12(1):1073, 2021.

[45] Laura Di Domenico, Chiara E Sabbatini, Pierre-Yves Boëlle, Chiara Poletto, Pascal Crépey, Juliette Paireau, Simon Cauchemez, François Beck, Harold Noel, Daniel Lévy-Bruhl, et al. Adherence and sustainability of interventions informing optimal control against the covid-19 pandemic. *Communications medicine*, 1(1):57, 2021.

[46] Laura Di Domenico, Chiara E Sabbatini, Giulia Pullano, Daniel Lévy-Bruhl, and Vittoria Colizza. Impact of january 2021 curfew measures on sars-cov-2 b. 1.1. 7 circulation in france. *Eurosurveillance*, 26(15):2100272, 2021.

[47] Michelangelo Diligenti, Marco Gori, and Claudio Sacca. Semantic-based regularization for learning and inference. *Artificial Intelligence*, 244:143–165, 2017.

[48] Michelangelo Diligenti, Soumali Roychowdhury, and Marco Gori. Integrating prior knowledge into deep learning. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 920–923. IEEE, 2017.

[49] D. K. Djakpo, Z. Wang, R. Zhang, X. Chen, P. Chen, and M. M. L. K. Antoine. Blood routine test in mild and common 2019 coronavirus (covid-19) patients. *Biosci Rep*, 40(8):BSR20200817, Aug 2020.

[50] Priya Donti, Brandon Amos, and J Zico Kolter. Task-based end-to-end model learning in stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 5484–5494, 2017.

[51] W. Duan, Z. Fan, P. Zhang, G. Guo, and X. Qiu. Mathematical and computational approaches to epidemic modeling: a comprehensive review. *Frontiers of Computer Science*, 9(5):806–826, 2015.

[52] Elad Eban, Mariano Schain, Alan Mackey, Ariel Gordon, Ryan Rifkin, and Gal Elidan. Scalable learning of non-decomposable objectives. In *Artificial Intelligence and Statistics*, pages 832–840. PMLR, 2017.

[53] L. Euler. Recherches générales sur la mortalité et la multiplication du genre humain. 1760.

[54] Mahdi Milani Fard, Kevin Canini, Andrew Cotter, Jan Pfeifer, and Maya Gupta. Fast and flexible monotonic functions with ensembles of lattices. In *Advances in Neural Information Processing Systems*, pages 2919–2927, 2016.

[55] J. Farooq and M. A. Bazaz. A novel adaptive deep learning model of covid-19 with focus on mortality reduction strategies. *Chaos, Solitons & Fractals*, 138:110148, 2020.

[56] Aaron Ferber, Bryan Wilder, Bistra Dilina, and Milind Tambe. Mipaal: Mixed integer program as a layer. *arXiv preprint arXiv:1907.05912*, 2019.

[57] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[58] Ferdinando Fioretto, Pascal Van Hentenryck, Terrence WK Mak, Cuong Tran, Federico Baldo, and Michele Lombardi. Lagrangian duality for constrained deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 118–135. Springer, 2020.

[59] Ferdinando Fioretto, Terrence WK Mak, and et al. A lagrangian dual framework for deep neural networks with constraints. *arXiv preprint arXiv:2001.09394*, 2020.

[60] Marc Fischer, Mislav Balunovic, Dana Drachsler-Cohen, Timon Gehr, Ce Zhang, and Martin Vechev. Dl2: Training and querying neural networks with logic. In *International Conference on Machine Learning*, 2019.

[61] Thomas Frerix, Matthias Nießner, and Daniel Cremers. Homogeneous linear inequality constraints for neural network activations. *arXiv preprint arXiv:1902.01785*, 2019.

[62] Gilles Gasso, Aristidis Pappaioannou, Marina Spivak, and Léon Bottou. Batch and online learning algorithms for nonconvex neyman-pearson classification. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):1–19, 2011.

[63] Giulia Giordano, Franco Blanchini, Raffaele Bruno, Patrizio Colaneri, Alessandro Di Filippo, Angela Di Matteo, and Marta Colaneri. A sidarthe model of covid-19 epidemic in italy. *ArXiv*, abs/2003.09861, 2020.

[64] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[65] Gabriel Goh, Andrew Cotter, Maya Gupta, and Michael P Friedlander. Satisfying real-world goals with dataset constraints. In *Advances in Neural Information Processing Systems*, pages 2415–2423, 2016.

[66] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT press, 2016.

[67] Stephen Gould, Richard Hartley, and Dylan Campbell. Deep declarative networks: A new hope, 2019.

[68] Marijke Grau, Lars Ibershoff, Jonas Zacher, Janina Bros, Fabian Tomschi, Katharina Felicitas Diebold, Hans-Georg Predel, and Wilhelm Bloch. Even patients with mild covid-19 symptoms after sars-cov-2 infection show prolonged altered red blood cell morphology and rheological parameters. *Journal of Cellular and Molecular Medicine*, 26(10):3022–3030, 2022.

[69] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer, 2005.

[70] Maya Gupta, Dara Bahri, Andrew Cotter, and Kevin Canini. Diminishing returns shape constraints for interpretability and regularization. In *Advances in neural information processing systems*, pages 6834–6844, 2018.

[71] Maya Gupta, Andrew Cotter, Jan Pfeifer, Konstantin Voevodski, Kevin Canini, Alexander Mangylov, Wojciech Moczydlowski, and Alexander Van Esbroeck. Monotonic calibrated interpolated look-up tables. *The Journal of Machine Learning Research*, 17(1):3790–3836, 2016.

[72] Schaeffer Hayden. Learning partial differential equations via data discovery and sparse optimization. *Proc. R. Soc. A.47320160446 20160446*, 2017.

[73] A Hernández-García and P König. Data augmentation instead of explicit regularization. *arXiv preprint arXiv:1806.03852*, 2018.

[74] Alex Hernández-García and Peter König. Further advantages of data augmentation on convolutional neural networks. In *International Conference on Artificial Neural Networks*. Springer, 2018.

[75] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[76] Johannes J.M.L. Hoffmann, Karin C.A.M. Nabbe, and Nicole M.A. van den Broek. Effect of age and gender on reference intervals of red blood cell distribution width (rdw) and mean red cell volume (mcv). *Clinical Chemistry and Laboratory Medicine (CCLM)*, 53(12):2015–2019, 2015.

[77] Ehsan Hosseini-Asl, Jacek M Zurada, and Olfa Nasraoui. Deep learning of part-based representation of data using sparse autoencoders with nonnegativity constraints. *IEEE transactions on neural networks and learning systems*, 27(12):2486–2498, 2015.

[78] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*, 2016.

[79] C. J. Huang, Y. H. Chen, Y. Ma, and P. H. Kuo. Multiple-input deep convolutional neural network model for covid-19 forecasting in china. *medRxiv*, 2020.

[80] C. J. Huang, Y. Shen, P. H. Kuo, and Y. H. Chen. Novel spatiotemporal feature extraction parallel deep neural network for forecasting confirmed cases of coronavirus disease 2019. *medRxiv*, 2020.

[81] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. Automated machine learning: Methods, systems, challenges. *Automated Machine Learning*, 2019.

[82] M. R. Ibrahim, J. Haworth, A. Lipani, N. Aslam, T. Cheng, and N. Christie. Variational-lstm autoencoder to forecast the spread of coronavirus across the globe. *medRxiv*, 2020.

[83] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 5308–5317, 2016.

[84] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, et al. *An introduction to statistical learning*, volume 112. Springer, 2013.

[85] Xiaowei Jia, Jared Willard, Anuj Karpatne, Jordan Read, Jacob Zwart, Michael Steinbach, and Vipin Kumar. Physics guided rnns for modeling dynamical systems: A case study in simulating lake temperature profiles. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 558–566. SIAM, 2019.

[86] H. Jo, H. Son, H. J. Hwang, and S. Y. Jung. Analysis of covid-19 spread in south korea using the sir model with time-dependent parameters and deep learning. *medRxiv*, 2020.

[87] Kadierdan Kaheman, Steven L Brunton, and J Nathan Kutz. Automatic differentiation to simultaneously identify nonlinear dynamics and extract noise probability distributions from data. *Machine Learning: Science and Technology*, 3(1):015031, 2022.

[88] F. Kamiran and T. Calders. Classifying without discriminating. In *2009 2nd International Conference on Computer, Control and Communication*, pages 1–6, 2009.

[89] Faisal Kamiran and Toon Calders. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33(1):1–33, 2012.

[90] Faisal Kamiran, Toon Calders, and Mykola Pechenizkiy. Discrimination aware decision tree learning. In *2010 IEEE International Conference on Data Mining*, pages 869–874. IEEE, 2010.

[91] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[92] Anuj Karpatne, William Watkins, Jordan S. Read, and Vipin Kumar. Physics-guided neural networks (PGNN): an application in lake temperature modeling. *CoRR*, abs/1710.11431, 2017.

[93] Samina Khalid, Tehmina Khalil, and Shamila Nasreen. A survey of feature selection and feature extraction techniques in machine learning. In *2014 Science and Information Conference*, pages 372–378. IEEE, 2014.

[94] Udayan Khurana, Horst Samulowitz, and Deepak Turaga. Feature engineering for predictive modeling using reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[95] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

[96] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017.

[97] A. Kunjir, D. Joshi, R. Chadha, T. Wadiwala, and V. Trikha. A comparative study of predictive machine learning algorithms for covid-19 trends and analysis. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3407–3412, 2020.

[98] Zhilu Lai, Charilaos Mylonas, Satish Nagarajaiah, and Eleni Chatzi. Structural identification with physics-informed neural ordinary differential equations. *Journal of Sound and Vibration*, 508:116196, 2021.

[99] Fabien Lauer and Gérard Bloch. Incorporating prior knowledge in support vector machines for classification: A review. *Neurocomputing*, 71(7-9):1578–1594, 2008.

[100] C. Lea, R. Vidal, A. Reiter, and G. D. Hager. Temporal convolutional networks: A unified approach to action segmentation. In G. Hua and H. Jégou, editors, *Computer Vision – ECCV 2016 Workshops*, pages 47–54, Cham, 2016. Springer International Publishing.

[101] Tao Li and Vivek Srikumar. Augmenting neural networks with first-order logic. *arXiv preprint arXiv:1906.06298*, 2019.

[102] Xiaodan Liang, Zhiting Hu, Hao Zhang, Liang Lin, and Eric P Xing. Symbolic graph reasoning meets convolutions. In *Advances in Neural Information Processing Systems*, pages 1853–1863, 2018.

[103] Isabella Locatelli, Bastien Trächsel, and Valentin Rousson. Estimating the basic reproduction number for covid-19 in western europe. *PLoS ONE*, 03 2021.

[104] Alfred J. Lotka. The relation between birth rate and death rate in a normal population and the rational basis of an empirical formula for the mean length of life given by william farr. *Publications of the American Statistical Association*, 16(123):121–130, 1918.

[105] Peter Y Lu, Samuel Kim, and Marin Soljačić. Extracting interpretable physical parameters from spatiotemporal systems using unsupervised learning. *Physical Review X*, 10(3):031056, 2020.

[106] Kenneth Marino, Ruslan Salakhutdinov, and Abhinav Gupta. The more you know: Using knowledge graphs for image classification. *arXiv preprint arXiv:1612.04844*, 2016.

[107] Giuseppe Marra, Michelangelo Diligenti, Francesco Giannini, Marco Gori, and Marco Maggini. Relational neural machines. *arXiv preprint arXiv:2002.02193*, 2020.

[108] Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, and Marco Gori. Integrating learning and reasoning with deep logic models. *arXiv preprint arXiv:1901.04195*, 2019.

[109] Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, and Marco Gori. Lyrics: a general interface layer to integrate logic inference and deep learning. In *ECML-PKDD 2019*, 2019.

[110] S. Mežnar, N. Lavrač, and B. Škrlj. Prediction of the effects of epidemic spreading with graph neural networks. In R. M. Benito, C. Cherifi, H. Cherifi, E. Moro, L. M. Rocha, and M. Sales-Pardo, editors, *Complex Networks & Their Applications IX*, pages 420–431, Cham, 2021. Springer International Publishing.

[111] Sebastian Mežnar, Nada Lavrač, and Blaž Škrlj. Transfer learning for node regression applied to spreading prediction. *arXiv preprint arXiv:2104.00088*, 2021.

[112] Rebecca Miao, Zhenyi Yang, and Valeriy Gavrishchaka. Leveraging domain-expert knowledge, boosting and deep learning for identification of rare and complex states. In *Journal of Physics: Conference Series*, volume 1207, page 012016. IOP Publishing, 2019.

[113] Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Adversarial sets for regularising neural link predictors. *ArXiv*, abs/1707.07596, 2017.

[114] Amina Mollaysa, Alexandros Kalousis, Eric Bruno, and Maurits Diephuis. Learning to augment with feature side-information. In *Asian Conference on Machine Learning*, pages 173–187, 2019.

[115] Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li F Fei-Fei, Josh Tenenbaum, and Daniel L Yamins. Flexible neural representation for physics prediction. In *Advances in neural information processing systems*, pages 8799–8810, 2018.

[116] N. Muralidhar, M. R. Islam, M. Marwah, A. Karpatne, and N. Ramakrishnan. Incorporating prior domain knowledge into deep neural networks. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 36–45, 2018.

[117] N Muralidhar, MR Islam, and et al. Incorporating prior domain knowledge into deep neural networks. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018.

[118] Fatemeh Nargesian, Horst Samulowitz, Udayan Khurana, Elias B Khalil, and Deepak S Turaga. Learning feature engineering for classification. In *IJCAI*, pages 2529–2535, 2017.

[119] Elaine O Nsoesie, John S Brownstein, Naren Ramakrishnan, and Madhav V Marathe. A systematic review of studies on forecasting the dynamics of influenza outbreaks. *Influenza and other respiratory viruses*, 8(3):309–316, 2014.

[120] Jared O'Leary, Joel A. Paulson, and Ali Mesbah. Stochastic physics-informed neural ordinary differential equations. *Journal of Computational Physics*, 468:111466, nov 2022.

[121] N. Oliver, E. Letouzé, H. Sterly, S. Delataille, M. De Nadai, B. Lepri, R. Lambiotte, R. Benjamins, C. Cattuto, V. Colizza, N. de Cordes, S. P. Fraiberger, T. Koebe, S. Lehmann, J. Murillo, A. Pentland, P. N. Pham, F. Pivetta, A. Ali Salah, J. Saramäki, S. V. Scarpino, M. Tizzoni, S. Verhulst, and P. Vinck. Mobile phone data and covid-19: Missing an opportunity?, 2020.

[122] George Panagopoulos, Giannis Nikolentzos, and Michalis Vazirgiannis. Transfer graph neural networks for pandemic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4838–4845, 2021.

[123] Deepak Pathak, Philipp Krahenbuhl, and Trevor Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1796–1804, 2015.

[124] Gaetano Perone. An arima model to forecast the spread and the final size of covid-2019 epidemic in italy. *MedRxiv*, pages 2020–04, 2020.

[125] Giulia Pullano, Laura Di Domenico, Chiara E Sabbatini, Eugenio Valdano, Clément Turbelin, Marion Debin, Caroline Guerrisi, Charly Kengne-Kuetche, Cécile Souty, Thomas Hanslik, et al. Underdetection of cases of covid-19 in france threatens epidemic control. *Nature*, 590(7844):134–139, 2021.

[126] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

[127] Luc De Raedt, Robin Manhaeve, Sebastijan Dumancic, Thomas Demeester, and Angelika Kimmig. Neuro-symbolic = neural + logical + probabilistic. 2019.

[128] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[129] Pranav Rajpurkar, Emily Chen, Oishi Banerjee, et al. Ai in health and medicine. *Nature Medicine*, 28:31–38, 2022.

[130] Sebastian Reich and Colin Cotter. *Probabilistic forecasting and Bayesian data assimilation*. Cambridge University Press, 2015.

[131] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[132] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[133] Kirstin Roster, Colm Connaughton, and Francisco A Rodrigues. Forecasting new diseases in low-data settings using transfer learning. *Chaos, Solitons & Fractals*, 161:112306, 2022.

[134] Samuel Rudy, Alessandro Alla, Steven L Brunton, and J Nathan Kutz. Data-driven identification of parametric partial differential equations. *SIAM Journal on Applied Dynamical Systems*, 18(2):643–660, 2019.

[135] Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017.

[136] Chiara Elisa Sabbatini, Giulia Pullano, Laura Di Domenico, Stefania Rubrichi, Shweta Bansal, and Vittoria Colizza. The impact of spatial connectivity on npis effectiveness. *medRxiv*, 2023.

[137] J. C. Santos and S. Matos. Analysing twitter and web queries for flu trend prediction. *Theoretical Biology and Medical Modelling*, 11(1):1–11, 2014.

[138] S. Sareen, S. K. Sood, and S. K. Gupta. Iot-based cloud framework to control ebola virus outbreak. *Journal of Ambient Intelligence and Humanized Computing*, 9:459 – 476, 2018.

[139] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[140] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[141] Luciano Serafini, Ivan Donadello, and Artur d'Avila Garcez. Learning and reasoning in logic tensor networks: theory and application to semantic image interpretation. In *Proceedings of the Symposium on Applied Computing*, pages 125–130, 2017.

[142] F. Shahid, A. Zameer, and M. Muneeb. Predictions for covid-19 with deep learning models of lstm, gru and bi-lstm. *Chaos, Solitons & Fractals*, 140:110212, 2020.

[143] S. Shastri, K. Singh, S. Kumar, P. Kour, and V. Mansotra. Time series forecasting of covid-19 using deep learning models: India-usa comparative case study. *Chaos, Solitons & Fractals*, 140:110227, 2020.

[144] Mohammad Shorfuzzaman and M Shamim Hossain. Metacovid: A siamese neural network framework with contrastive loss for n-shot diagnosis of covid-19 patients. *Pattern recognition*, 113:107700, 2021.

[145] Mattia Silvestri, Federico Baldo, Eleonora Misino, and Michele Lombardi. An analysis of universal differential equations for data-driven discovery of ordinary differential equations. *arXiv preprint arXiv:2306.10335*, 2023.

[146] Mattia Silvestri, Michele Lombardi, and Michela Milano. Injecting domain knowledge in neural networks: a controlled experiment on a constrained problem. *arXiv preprint arXiv:2002.10742*, 2020.

[147] L. Simko, R. Calo, F. Roesner, and T. Kohno. Covid-19 contact tracing and privacy: Studying opinion and preferences, 2020.

[148] Nathaniel R Smilowitz, Dennis Kunichoff, Michael Garshick, Binita Shah, Michael Pillinger, Judith S Hochman, and Jeffrey S Berger. C-reactive protein and clinical outcomes in patients with covid-19. *European heart journal*, 42(23):2270–2279, 2021.

[149] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[150] Russell Stewart and Stefano Ermon. Label-free supervision of neural networks with physics and domain knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[151] Aukse Stravinskienė, Saulius Gudas, and Aiste Dabrilaite. Decision tree algorithms: Integration of domain knowledge for data mining. In *International Conference on Business Information Systems*, pages 13–24. Springer, 2012.

[152] Naoya Takeishi and Yoshinobu Kawahara. Regularizing generative models using knowledge of feature dependence. 2019.

[153] Geoffrey G Towell and Jude W Shavlik. Knowledge-based artificial neural networks. *Artificial intelligence*, 70(1-2):119–165, 1994.

[154] T. P. Velavan and C. G. Meyer. The covid-19 epidemic. *Tropical medicine & international health*, 25(3):278, 2020.

[155] S. R. Venna, A. Tavanaei, R. N. Gottumukkala, V. V. Raghavan, A. S. Maida, and S. Nichols. A novel data-driven model for real-time influenza forecasting. *IEEE Access*, 7:7691–7701, 2019.

[156] João Vieira and Cláudia Antunes. Decision tree learner in the presence of domain knowledge. In *Chinese Semantic Web and Web Science Conference*, pages 42–55. Springer, 2014.

[157] Marin Vlastelica, Anselm Paulus, Vít Musil, Georg Martius, and Michal Rolínek. Differentiation of blackbox combinatorial solvers. *arXiv preprint arXiv:1912.02175*, 2019.

[158] Khuong Vo, Dang Pham, Mao Nguyen, Trung Mai, and Tho Quan. Combination of domain knowledge and deep learning for sentiment analysis. In *International Workshop on Multi-disciplinary Trends in Artificial Intelligence*, pages 162–173. Springer, 2017.

[159] Laura von Rueden, Sebastian Mayer, and et al. Informed machine learning – a taxonomy and survey of integrating knowledge into learning systems. *arXiv preprint arXiv:1903.12394v2*, 2020.

[160] Jason Wang and Luis Perez. The effectiveness of data augmentation in image classification using deep learning. *Convolutional Neural Networks Vis. Recognit*, page 11, 2017.

[161] L. Wang, J. Chen, and M. Marathe. Tdefsi: Theory-guided deep learning-based epidemic forecasting with synthetic information. *ACM Trans. Spatial Algorithms Syst.*, 6(3), April 2020.

[162] Po-Wei Wang, Priya L Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. *arXiv preprint arXiv:1905.12149*, 2019.

[163] Serena Wang and Maya Gupta. Deontological ethics by monotonicity shape constraints. *arXiv preprint arXiv:2001.11990*, 2020.

[164] Y. Wang, C. Xu, Y. Li, W. Wu, L. Gui, J. Ren, and S. Yao. An advanced data-driven hybrid model of sarima-nnnar for tuberculosis incidence time series forecasting in qinghai province, china. *Infection and drug resistance*, 13:867, 2020.

[165] Y. Wang, C. Xu, S. Zhang, Z. Wang, L. Yang, Y. Zhu, and J. Yuan. Temporal trends analysis of tuberculosis morbidity in mainland china from 1997 to 2025 using a new sarima-narnnx hybrid model. *BMJ Open*, 9(7), 2019.

[166] W. Wei, J. Jiang, H. Liang, L. Gao, B. Liang, J. Huang, N. Zang, Y. Liao, J. Yu, J. Lai, F. Qin, J. Su, L. Ye, and H. Chen. Application of a combined model with autoregressive integrated moving average (arima) and generalized regression neural network (grnn) in forecasting hepatitis incidence in heng county, china. *PLOS ONE*, 11(6):1–13, 06 2016.

[167] Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1658–1665, 2019.

[168] Bryan Wilder, Eric Ewing, Bistra Dilkina, and Milind Tambe. End to end learning and optimization on graphs. In *Advances in Neural Information Processing Systems*, pages 4674–4685, 2019.

[169] Y. Wu, Y. Yang, H. Nishiura, and M. Saitoh. Deep learning for epidemiological predictions. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 1085–1088, New York, NY, USA, 2018. Association for Computing Machinery.

[170] Hao Xu, Haibin Chang, and Dongxiao Zhang. Dl-pde: Deep-learning based data-driven discovery of partial differential equations from discrete and noisy data. *arXiv preprint arXiv:1908.04463*, 2019.

[171] Jingyi Xu, Zilu Zhang, and et al. A semantic loss function for deep learning with symbolic knowledge. In *Proceedings of the 35th ICML*, 2018.

[172] Xie Yaqi, Ziwei Xu, Kuldeep S Meel, Mohan Kankanhalli, and Harold Soh. Embedding symbolic knowledge into deep networks. In *Advances in Neural Information Processing Systems*, pages 4235–4245, 2019.

[173] Seungil You, David Ding, Kevin Canini, Jan Pfeifer, and Maya Gupta. Deep lattice networks and partial monotonic functions. In *Advances in neural information processing systems*, pages 2981–2989, 2017.

[174] Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, 2019.

[175] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):11, 2019.