Machine Learning Algorithms for the Calibration of Epidemiological Compartmental Models

Application to the Italian COVID-19 outbreaks

Tutor : Professor Luca **Dedè**

Alessandro **Del Vitto**

Michele **Di Sabato**

Academic Year 2020-2021

Numerical Analysis for Partial Differential Equations

Contents

Abstract

One of the many challenges proposed by the ongoing COVID-19 pandemic is to explain the spread of the disease through suitable equations and capture the epidemic's features through the parameters of the mathematical model. However some of these values cannot be pre-determined nor measured precisely, hence other estimation procedures are needed. Machine Learning has been extensively used as a parametric and data-based model identification technique. In this report we use Machine Learning with the purpose of testing and exploiting the learning capability of Artificial Neural Networks, applied to noisy, real-world data, to obtain a better estimation of the unknown parameters of epidemiological compartmental models. Motivated by the effectiveness of Machine Learning, we used the trained neural network to estimate the basic reproduction number of the COVID-19 epidemic and the contagion rate of the disease. We conclude that a neural-network-informed calibration method provides a reliable result, when compared with standard techniques.

1 Introduction

Compartmental epidemiological models for the description of the COVID-19 pandemic rely on different parameters whose values cannot be set *a priori* nor determined exactly. As explained in [Nic21], these parameters are usually empirically tuned in a procedure called "calibration" where the model is turned inside-out and solved onto a known scenario by minimizing the mathematical distance between the model solution and the real one. In this paper we apply Machine Learning methods in order to obtain an alternative calibration procedure, which does not require any measure of the desired parameter in the calibration period. Thanks to the application of Artificial Neural Networks in data-driven model identification, for which we refer to [Fra19], we are able to obtain a reliable estimation by determining the starting point for the calibration procedure exploiting the network knowledge. This method requires only a sufficiently large and well diversified training set with which to train the Neural Network and the experience of the designer in the choice of the hyperparameters for the learning process.
We first explore the data driven machine learning framework of model identification to understand how to consistently obtain a well-trained Neural Network, that is to obtain a robust learning procedure. We then apply our calibration technique to the SEIRD model calibrating one of the most important parameters of the disease, the transmission rate β, which is strictly related to the nowadays well-known reproduction number R(t). We then proceed to show how in a scenario in which it is not possible to start from a relatively accurate measure of the parameter β, our Neural Network calibration could be a reliable alternative. In the training we exploit the data available at [Con] the online repository published by the Italian Dipartimento di Protezione Civile e Consiglio dei Ministri and we learn the law standing behind an approximation of the basic reproduction number of the pandemic defined in [Bat], namely $\mathcal{R}^*$.

2 Model Identification with Neural Networks

Two of the many applications of Machine Learning algorithms are data based Model Order Reduction (MOR) and data based Model Identification. These applications are mostly exploited by Ordinary Differential Equations (ODEs) or Partial Differential Equations (PDEs) driven systems and require a collection of input-output pairs, that are either generated through a High Fidelity (HF) model or collected through experiments. As explained in [Fra19] the benefit of these models relies on the fact that in many applications, where numerous simulations of different scenarios are needed (e.g. medicine, meteorology), the computational cost of solving the HF model becomes overwhelming, therefore the implementation of a reduced or surrogate model might be a useful alternative.

2.1 From HF model to Reduced Order model:

The following is a short introduction to the mathematical framework of Model Order Reduction and Model Identification. For a throughout explanation we refer to [Fra19]. Given a time dependent input $\mathbf{u}(t) \in \mathbb{R}^{N_u}$ and considering $\mathbf{y}(t) \in \mathbb{R}^{N_y}$ as the output, the general formulation of the HF model is the following:

$$\begin{cases} \dot{\mathbf{X}}(t) = \mathbf{F}(\mathbf{X}(t), \mathbf{u}(t)) \qquad t \in (0, T] \\ \mathbf{X}(0) = \mathbf{X}_0 \\ \mathbf{y}(t) = \mathbf{G}(\mathbf{X}(t)) \qquad t \in (0, T] \end{cases} \tag{1}$$

with $\mathbf{F} : \mathbb{R}^N \times \mathbb{R}^{N_u} \to \mathbb{R}^N$, where $\mathbf{G} : \mathbb{R}^N \to \mathbb{R}^{N_y}$ and $\mathbf{X}(t) \in \mathbb{R}^N$ represents the state of the HF model[1].

The corresponding Reduced Order model is:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \qquad t \in (0, T] \\ \mathbf{x}(0) = \mathbf{x}_0 \\ \tilde{y}(t) = \mathbf{g}(\mathbf{x}(t)) \qquad t \in (0, T] \end{cases} \tag{2}$$

with $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^{N_u} \to \mathbb{R}^n$, where $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^{N_y}$ and $\mathbf{x}(t) \in \mathbb{R}^n$ represents the state of the Reduced Model. The reduction is achieved thanks to the fact that $n <<< N$, meaning that that the number of states used in (2) is much lower than the states in (1).

Remark. *Notice that the knowledge of the entire state $X(t)$ is not essential, since in many applications the only relevant information is the map that links the output with the input, which we will call $\boldsymbol{\varphi}$.*

Summary and goals:

Machine Learning can be used to reconstruct the internal state $\mathbf{X}$(t) of a system like (1) based on its reduced representation $\mathbf{x}$(t) (without the possibility of observing $\mathbf{X}$(t) itself) and to find a model for the dynamics of $\mathbf{x}$(t).
This framework serves the following purposes:

1. Learning mathematical models based only on input-output pairs
2. Building surrogate models, that allow for fast evaluations
3. Reducing the dimensionality of the original HF model

Throughout our project, we mainly used Machine Learning to achieve the first goal.

[1] $\mathbf{X}(t) \in \mathbb{R}^N$ for finite dimensional models (e.g. ODEs), $\mathbf{X}(t)$ infinite dimensional for infinite dimensional models (e.g. PDEs)

It is important to notice that the reconstruction of the system state through $\mathbf{x}$(t) is *only* instrumental to the reconstruction of the input-output map $\boldsymbol{\varphi}$: $\mathbf{u} \longrightarrow \mathbf{y}$.
To better understand how a Machine Learning algorithm learns an input-output map, it is necessary to clarify some definitions and conventions that will be followed throughout this report. The following is a broad description of the mathematical framework on which our project is based.

Let $\mathcal{U}$ and $\mathcal{Y}$ be subspaces of $\mathbb{R}^{N_u}$ and $\mathbb{R}^{N_y}$ respectively and such that $\mathbf{u} : [0,T] \to \mathcal{U}$ and $\mathbf{y} : [0,T] \to \mathcal{Y}$, the collection of all the possible maps that link the input to the output is defined as:

$$\begin{aligned}\boldsymbol{\Phi} \;:= \big\{ & \boldsymbol{\varphi} : \mathcal{U} \to \mathcal{Y} \quad \text{such that} \\ & \varphi(u) = y \quad \text{and} \\ & \exists\; y_0 \in \mathcal{Y} \text{ such that } \forall u \in \mathcal{U} \;\; (\varphi u)(0) = y_0 \quad \text{and} \\ & \forall u_1,\; u_2 \in \mathcal{U} \;\; \forall t^* \in [0,T] \quad u_1|_{[0,t^*]} = u_2|_{[0,t^*]} \implies (\varphi u_1)|_{[0,t^*]} = (\varphi u_2)|_{[0,t^*]} \big\}\end{aligned}$$

The learning algorithm is the following:

1. collect (or compute using the HF model, when available) N_s couples $\{\hat{u}_j, \hat{y}_j\}_{j=1,2,...,N_s}$ of input-output pairs, which will be called *samples*
2. define a cost functional $J(\varphi) = \frac{1}{2}\sum_{j=1}^{N_s} \int_0^T (\hat{y}_j(t) - (\varphi \hat{u}_j)(t))^2 dt$
3. choose an appropriate subset of candidate models $\widehat{\boldsymbol{\Phi}} \subseteq \boldsymbol{\Phi}$
4. find $\boldsymbol{\varphi}^* = \text{argmin}_{\;\boldsymbol{\varphi} \in \widehat{\boldsymbol{\Phi}}} J(\theta)$

One of the most crucial steps of the previously explained procedure, is the definition of $\widehat{\boldsymbol{\Phi}}$, as this choice is problem-dependent.
The mathematical formulation of the problem is:

$$\begin{cases} min_{\mathbf{f} \,\in\, \widehat{\mathcal{F}},\; \mathbf{g} \,\in\, \widehat{\mathcal{G}},\; \mathbf{x}_0 \,\in\, \widehat{\mathcal{X}}} \displaystyle\sum_{j=1}^{N_s} \int_0^T (\hat{y}_j(t) - \mathbf{g}(\mathbf{x}_j(t)))^2 dt \\ \dot{\mathbf{x}}_j = \mathbf{f}(\mathbf{x}_j(t), \hat{\mathbf{u}}_j(t)), \quad t \;\in\; [0,T], \quad j = 1,2,...,N_s \\ \mathbf{x}_j(0) = \mathbf{x}_0, \quad j = 1,2,...,N_s \end{cases} \tag{3}$$

Where $\widehat{\mathcal{X}}$ is the space containing all possible initial values (e.g. $\mathbb{R}^N$) and $\widehat{\mathcal{F}}$ and $\widehat{\mathcal{G}}$ are suitable approximations of the spaces which the functions $\mathbf{f}$ and $\mathbf{g}$ belong to.
One possible issue regarding (3) is that the state space representation $(\mathbf{f}, \mathbf{g}, \mathbf{x}_0)$ corresponding to the I/O representation ($\boldsymbol{\varphi}$), is <u>not</u> unique. Therefore, to address this issue, there are two possible variations of (3):

1. **output-outside-the-state**: $\widehat{\mathcal{X}} = \{\mathbf{0}\}$ (the output is a function of the state, but it is not a part of the state, since $\mathbf{x}_0$ is set to $\mathbf{0}$).
2. **output-inside-the-state**: $\widehat{\mathcal{X}} = \{(\mathbf{y}_0^T, \mathbf{0}^T)^T\}$ and $\widehat{\mathcal{G}} = \{\boldsymbol{\pi}^{N_y}\}$ (meaning that the first N_y components of the reduced state vector $\mathbf{x}(t)$ (called $\boldsymbol{\pi}^{N_y}$) coincide exactly with the output variables, hence the name "output-*inside*-the-state").

These two equivalent variations generate two different formulations of (3):

1. output-outside-the-state:

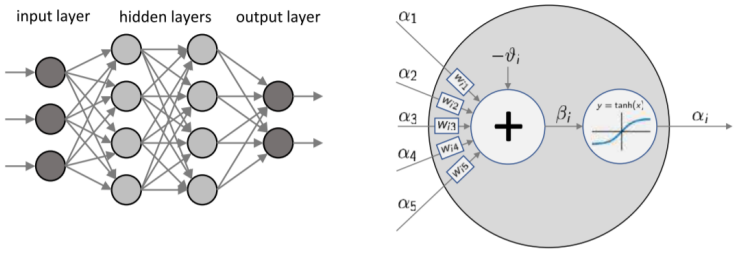
$$\begin{cases} min_{\mathbf{f} \in \widehat{\mathcal{F}},\, \mathbf{g} \in \widehat{\mathcal{G}}} \sum_{j=1}^{N_s} \int_0^T (\hat{y}_j(t) - \mathbf{g}(\mathbf{x}_j(t)))^2 dt \\ \dot{\mathbf{x}}_j = \mathbf{f}(\mathbf{x}_j(t), \hat{\mathbf{u}}_j(t)), \quad t \in [0,T], \quad j = 1,2,...,N_s \\ \mathbf{x}_j(0) = \mathbf{0}, \quad j = 1,2,...,N_s \end{cases} \tag{4}$$

2. output-inside-the-state:

$$\begin{cases} min_{\mathbf{f} \in \widehat{\mathcal{F}}} \sum_{j=1}^{N_s} \int_0^T (\hat{y}_j(t) - \boldsymbol{\pi}^{N_y}(\mathbf{x}_j(t)))^2 dt \\ \dot{\mathbf{x}}_j = \mathbf{f}(\mathbf{x}_j(t), \hat{\mathbf{u}}_j(t)), \quad t \in [0,T], \quad j = 1,2,...,N_s \\ \mathbf{x}_j(0) = \mathbf{x}_0, \quad j = 1,2,...,N_s \end{cases} \tag{5}$$

Observation:
the two variations are equivalent, since the state variables $\mathbf{x}$ are only auxiliary variables and don't necessarily have a physical interpretation. Throughout our project we used the output-inside-the-state version.

2.2 Neural Networks



(a) Feed-forward ANN (b) Action of the i-th neuron

Figure 1: Scheme of a feed-forward ANN with two hidden layers (a) and of a generated neuron (b). Credits: [Fra19]

Artificial Neural Networks consist in a number of simple processing units (called *neurons*), each one incorporating a linear and nonlinear application, interconnected to form a complex network.
It is possible to prove that any ANN with a single hidden layer can approximate with arbitrarily small error any continuous function on a compact set, provided that a sufficient number of hidden neurons are employed. Thanks to this feature, $\widehat{\mathcal{F}}$ and $\widehat{\mathcal{G}}$ (which are the spaces that define $\widehat{\boldsymbol{\Phi}}$) are chosen as subsets of the space of functions represented by ANNs. It can be shown that models represented by ANNs are universal approximators of the class of models described by a system of ODEs.
Neural Networks approximate problems such as (5) and (4) by

- parametrizing the functions $\mathbf{f}$ and $\mathbf{g}$ through the parameter vectors $\boldsymbol{\mu} \in \mathbb{R}^{N_f}$ and $\boldsymbol{\nu} \in \mathbb{R}^{N_g}$
- defining the discrete counterpart for the cost functional $J(\theta)$
- finding the set of parameters $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ that minimizes the new cost function

Exploiting the approximation of the Neural Networks, the new problem is the following:

$$\begin{cases} min_{(\boldsymbol{\mu},\boldsymbol{\nu})\in\mathbb{R}^{N_f+N_g}} \frac{1}{2}\Delta t \sum_{j=1}^{N_s}\sum_{k=0}^{M_j-1} |\mathbf{y}_j^k - \mathbf{g}(\mathbf{x}_j^k;\boldsymbol{\nu})|^2 \\ \dot{\mathbf{x}}_j^{k+1} = \mathbf{x}_j^k + \Delta t \mathbf{f}(\mathbf{x}_j^k, \mathbf{u}_j^k; \boldsymbol{\mu}), \quad k = 0, 1, ..., M_j - 1, \quad j = 1, 2, ..., N_s \\ \mathbf{x}_j^0 = \mathbf{0}, \quad j = 1, 2, ..., N_s \end{cases} \tag{6}$$

For more details about problem (6) see [Fra19], section (3.1).
It is important to notice that once the ANN has been trained (i.e. when the minimisation has been completed and the best choice for $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ has been made), the reduced model is available in the form of problem (2).
Finally, there are a lot of options when it comes to the minimization of the discrete cost function: throughout our project we used the Levenberg-Marquardt algorithm.

2.3 Compartmental Epidemiological Models

Compartmental Epidemiological Models are a family of mathematical models widely used in describing the evolution of a disease: the ODEs that constitute these types of models require the population under study to be divided into a fixed number of categories, according to their status. An individual that belongs to a certain compartment might move to a different one: for example if the categories are Susceptible individuals (S), Infected individuals (I) and Recovered individuals (R), a person who has not yet contracted the disease (called for this reason susceptible) will be transferred to the compartment "I" if or when they encounter an infected individual. Then, according to the specific characteristics of the disease, they might improve their condition, moving on to the category "R". Because of the names of the categories, the previously described dynamics will be well explained by a SIR model, which is indeed the simplest compartmental epidemiological model. As it is clear, there are many different variations of these models, that depend on the nature of the disease under study[2].
The choice of the model to use to describe the spread of a disease is problem-specific: the SIR model, for example, might be considered inaccurate to describe the ongoing COVID-19 pandemic, since it does not take into account relevant categories, such as hospitalized individuals (recently the Italian government has adopted this information to address the state of the pandemic on a local and regional level), undetected infected individuals (people who have contracted COVID, but are not registered as infected individuals yet) or isolated individuals (if a person is under quarantine, their infection rate is much lower: the model needs to include this information). For these reasons, more accurate models have been recently developed and used to try to keep the COVID-19 pandemic under control. For our project we studied the SUIHTER model, introduced in [Nic21], but, on the implementation point of view, we mainly dealt with the SEIRD model, which will be later described in section (4).

Calibration:
One common denominator for most of the compartmental models is that the corresponding ODEs depend on many parameters: For example, regarding the SIR model:

$$\begin{cases} \frac{dS}{dt} = -\frac{\beta IS}{N} \qquad t \in (0, T] \\ \frac{dI}{dt} = \frac{\beta IS}{N} - \gamma I \qquad t \in (0, T] \\ \frac{dR}{dt} = \gamma I \qquad t \in (0, T] \end{cases} \tag{7}$$

[2]To cite some of them: SEIRD, SUIHITER, SIDARTHE and so on. See [Nic21].

The parameters β and γ represent, respectively, the average number of contacts per person per time (i.e. the transmission rate) and the removal or the recovery rate. The value of these parameters strongly influences the values of the states S, I and R. To be more specific, Figure 2 shows what happens to the three states of the model, if the parameters are sampled 200 times randomly from a uniform distribution between $[0.06, 0.18]$ for γ and $[0.25, 0.35]$ for β:

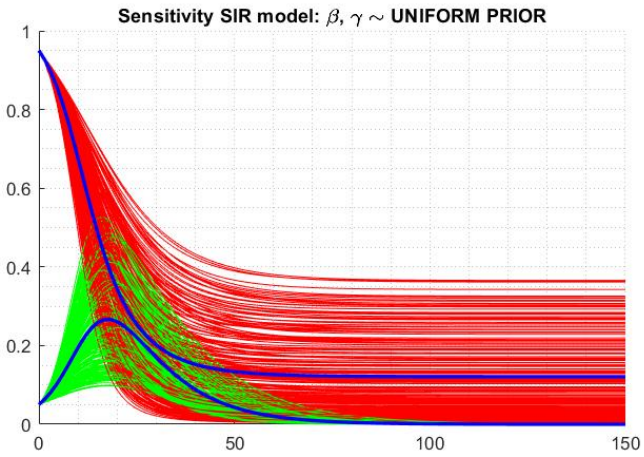


Figure 2: States S and I, when γ and β are sampled from uniform priors. The thick blue lines represent the mean curves.

Even though the intervals for the distributions are quite short, the solution of the equations (7) varies excessively. More complex models, such as SEIRD, involve many more parameters in the equations (since there are many more states), thus one of the main issues when dealing with compartmental models is to tune the values of the parameters of the model itself, so that the values of the compartments, computed as the solution of the model's equations, are similar to real world data (i.e. the measured values of the compartments). The issue of calibration will be addressed in Section 4.

Basic Reproduction Number:
As defined in [San], the Basic Reproduction Number, also known as $\mathcal{R}_0$, is the number of secondary infections caused by a single infectious, introduced into a wholly susceptible population over the course of the infection of this single infected individual[3]. Practically speaking, $\mathcal{R}_0$ represents the average number of susceptible individuals who will be infected by an individual who already has the disease: if $\mathcal{R}_0 \geq 1$, then the number of infected individuals will become larger and larger, otherwise the spread will be eventually under control.
There are many ways to compute the basic reproduction number. In particular, four of these techniques require to model the contagion based on statistical distributions attributed to the probability of contracting COVID and the duration of the contagion period. The names of these methods are:

1. Bettencourt-Ribeiro-Systrom
2. Wallinga-Teunis
3. a modified version of 2. developed by the Robert Koch Institute
4. a modified version of 2. developed by the Imperial College

[3]assuming stable conditions of the epidemic and constant contact's frequency. If this requirement is not fulfilled, then it is better to rely on the quantity called $\mathcal{R}(t)$.

Nonetheless, as shown in [Bat], these four methods produce conflicting outputs, which becomes problematic when the value of $\mathcal{R}(t)$ or $\mathcal{R}_0$ is used to asses the level of danger and to decide to implement lockdowns or not. Therefore, Professor Roberto Battiston, the author of [Bat], proposes a simpler calculation of the basic reproduction index, which is based on the solution of the ODEs, governing the SIR model:

$$\begin{aligned} \mathcal{R}_0^* &= \frac{\frac{1}{I(t)}\frac{dI}{dt}(t) + \gamma}{\gamma} = \frac{\beta r}{\gamma} \\ &\beta\text{: contagion rate} \\ &r\text{: contacts' number in a unit of time} \\ &\gamma\text{: healing rate} \\ &I(t)\text{: number of Infected individuals, computed solving the SIR equations} \end{aligned} \tag{8}$$

It is necessary to report other formulas that, together with (8), could be used to compute the Basic Reproduction Number:

- from the SEIRD model:

$$\begin{aligned} &\mathcal{R}_0 = \gamma\beta\frac{N}{S(t)} \\ &\beta\text{: contagion rate} \\ &\gamma\text{: death rate} \\ &N\text{: population of Italy} \\ &S(t)\text{: number of Susceptible individuals} \end{aligned} \tag{9}$$

- from the SUIHTER model:

$$\mathcal{R}_0 = \frac{\beta_U}{r_1} + \frac{\delta}{r_1}\left(\frac{\beta_I(r_3r_4 - \theta_T\omega_H) + \beta_H\omega_I r_4}{r_2r_3r_4 - r_4\theta_H\omega_I - r_2\theta_T\omega_H}\right) \tag{10}$$

(for more details on the parameters found in this formula, see [Nic21])

Two issues arise when using (8), (9) and (10): the first one is the fact that the parameters that appear in the formulas are often not known *a priori* (this is the case for COVID-19), so they would need to be estimated. Secondly, $I(t)$ and $S(t)$[4] are either the output of the HF model, or they are the measured, real-world data, which are obviously affected by noise. Particularly for the COVID-19 pandemic, the measurement of the number of infected people has been the topic of discussion in the media for a long time, since it is strongly influenced by the different efforts that each Italian region employs to test its inhabitants and by the false positive tests. Moreover, there have been many cases of people who, although being positive to COVID-19, were not regarded as Infected individuals, due to problems in the monitoring of the disease.

These measuring issues call for more objective and reliable ways to compute the basic reproduction number. Throughout the first part of our project, our attempt is to use the data-driven model identification techniques for ANNs mentioned before and explained in [Fra19] to learn the law standing beneath this parameter. The relevance of this task is emphasized by the fact that the Italian government used this value to establish lockdowns, to change restrictions on a national or regional level and assess the state of the pandemic.

[4] notice that in formula (9), $S(t) = N - I(t)$.

3 Learning of the Basic Reproduction Number

As already highlighted in Section 2.2 the basic reproduction number, or $\mathcal{R}^*$, introduced in [Bat], is a fundamental quantity for the description of a pandemic like the ongoing COVID-19. We recall that it is defined as the average number of people that a positive individual infects throughout his whole illness time. This parameter provides primary information about the evolution of the disease, mainly whether it is growing in size or, alternatively, scaling down. From this definition and considering the lack of information of a real world scenario it is immediately clear that it is not possible to determine it in his exact value. Relevant simplifications must be made. This is what has been done with mathematical models like compartmental epidemiological ones. They do not consider some aspects of the pandemic and try to mimic the evolution of the disease, reproducing its most relevant traits. They can be exceptionally precise in their approximation of the basic reproduction number but still the knowledge of the real world model could provide relevant help in the management of the worldwide emergency. With this aim our attempt is to implement a Neural Network scheme, as explained in [Fra19], that fed with the available data taken from [Con] and [Sta] would be able to learn and understand the true law, through which this quantity is determined.

In this section we are going to highlight and provide a throughout explanation of all the main passages that we followed in order to achieve a robust learning process. In particular: we will explain and analyse the data available in the repository [Con], we will comment the data on the $\mathcal{R}^*$ itself, we will explain how we have interacted with the library [Reg] and finally we will present the results of the three different types of tests that we have made.

Types of Tests:

1. *Training Tests* : to achieve a robust learning process
2. *Prediction Tests* : to use the Network to make predictions
3. *"What If" Analysis Tests* : to use the Network for analysis of past decisions

We underline that the main objective of our project was to develop a robust training process in order to apply the NN to the calibration of epidemiological compartmental models. For this reason we have been mainly concerned with the first type of tests, the other two were just the result of our exploring of all the different possible applications of ANNs.

3.1 The data

As soon as the emergency of the pandemic became clear to everyone an online repository ([Con]), was organized to store the most amount of information possible in order to help the description and the study of the disease. These data are available to everyone and they are continuously updated at the end of each day. They proved to be a very useful source of information but, as everything that is related to data acquisition, they are not without flaws. The main part of the repository consists in two different kinds of data, the first one is the information coming from the single regions and the second is the information about the whole country. We exploited both of them, for the first task - the learning part - mainly the local data, whereas the other was used in the second part for the calibration of the SEIRD model. Out of all the data available from [Con], we chose to collect only the quantities that could be relevant for the learning of the $\mathcal{R}^*$ and we discarded some of them for reliability issues. In the following list we are going to analyze and comment the relevant data.

- **Infectious**. The number of infectious individuals (Figure 17) is a very important quantity to describe the evolution of a pandemic but on the other end it is also one of the most difficult to obtain and to rely on. We want to underline that even tough the first wave of COVID-19 caught us completely unprepared and had a major impact, the number of people that were detected

with the disease is significantly lower for every region in the first wave with respect to the second one. This may be explained by the fact that the epidemic was not well monitored, especially at the start. Apart from some days of missing data and some steep changes that are due to corrections of past errors in the information collection procedure, we can say that there where no other major problems.

- **Hospitalized**. The number of hospitalized individuals (Figure 18) together with the number of intensive care unit patients can be viewed as the main concerns of the emergency. This is why we chose to include these information in the dataset.
- **Intensive Care**. This quantity (Figure 20) and the previous one enjoy the big advantage of being reliable in their measure. Note that, being related, they present a very similar behaviour.
- **Severity of Restriction**. This is an index (Figure 19) in the range $[0, 1]$ which represents the level at which the Italian government has set the restrictions. It gets close to zero when the restrictions get higher and it gets close to one whenever people are allowed to go out freely or not to wear a mask, as it was in the pre-pandemic. For instance from the first days of March to the 04/05/2020 it reached its lowest value since the whole Italy was on complete lockdown and it reached its highest peak during the following summer, when restrictions became weaker. It is also possible to notice that in the 3rd of November 2020 started the period in which regions were considered independently and divided in three different levels of restrictions: "Red","Orange" and "Yellow". Up until that day, the rules were set for Italy in its entirety, that is why all the twenty one graphs coincide. This quantity can be thought and it may be called as an index of the number of encounters that people make in their day to day lives, but we have stick to the name "Severity of restrictions" despite the counter-intuitive interpretation.
- **Deceased**. Another controversial value is the number of deceased individuals (see Figure 21). Differences in the criteria trough which a death is declared *due to COVID* make the counts in the different Regions non uniform and difficult to compare.
- **COVID Tests**. The number of tests that are performed each day is a considerable source of information on the monitoring of the pandemic and therefore on the reliability of many of the other data. One of the most important parameters is the one that indicates the percentage of the tests that turn out to be positive, but even the number of tests alone is enough to understand whether the information are collected in an complete way or whether some information is missing. From the plot (22) it is clear that in the first wave there was a monitoring problem and that the actual number of infectious, or even deceased, could be greater. From the technical point of view, this signal presents a weekly trend since on weekends the number of performed COVID tests is usually smaller than during working days. Some values are negative: this is probably due to errors in the data collection, that were then corrected readjusting the cumulative number of tests.

We also collected from [Con] data on the day-to-day variation of the number of positive individuals in order to check whether the training would be better if we fed the network with the variation instead of the cumulative number. As expected, the learning process did not show significant differences since the amount of information that it was receiving was, in principle, the same. On the other hand, we noticed that the network reflected in its estimated output the quick variation behaviour of the new input argument and this is why we chose to continue with the monotone signal given by the *total* number of infectious.

3.2 The output

$\mathcal{R}^*$ is the quantity that we wish to learn and so, as usual in black box model identification algorithms, it is necessary to have data about the output of the system gathered in a training set. These are the data with respect to which the Neural Network optimizes itself, rearranging its parameters in order to match *these* results.

The data about $\mathcal{R}^*$ are obtained in every single region through a simple approximated computation (see [Bat]) and inconsistencies in the values are not rare. The Figure 3 depicts the evolution of $\mathcal{R}^*$ for all the twenty-one Italian regions for the whole period that we took into consideration. First of all, it is very easy to observe the wavy behaviour of the pandemic from the shape of the curve. The values stand significantly above one mainly in two periods, the spring and the fall of 2020. Secondly, it is clear that there happen to be periods of time where $\mathcal{R}^*$ goes below zero and becomes negative. From the conceptual point of view this is a problem, considering the definition of $\mathcal{R}^*$, but since the quantity is derived on a day to day basis from other inconsistent data, it is possible to understand how these negative values are generated. The third and final point is that the $\mathcal{R}^*$ seems to be much greater in the first weeks of the pandemic with respect to the rest of the year. This can be explained as the simultaneous effect of the lack of preparation to face the emergency and the initial difficulties in gathering information in a truthful and systematic way. Moreover trough a more precise inspection, it is possible to see that the curves are not continuous but have a few missing values. Furthermore, note that the problem of negative values did not require any correction since it was a NN task to figure out that those were just inconsistencies and not features of the signal.

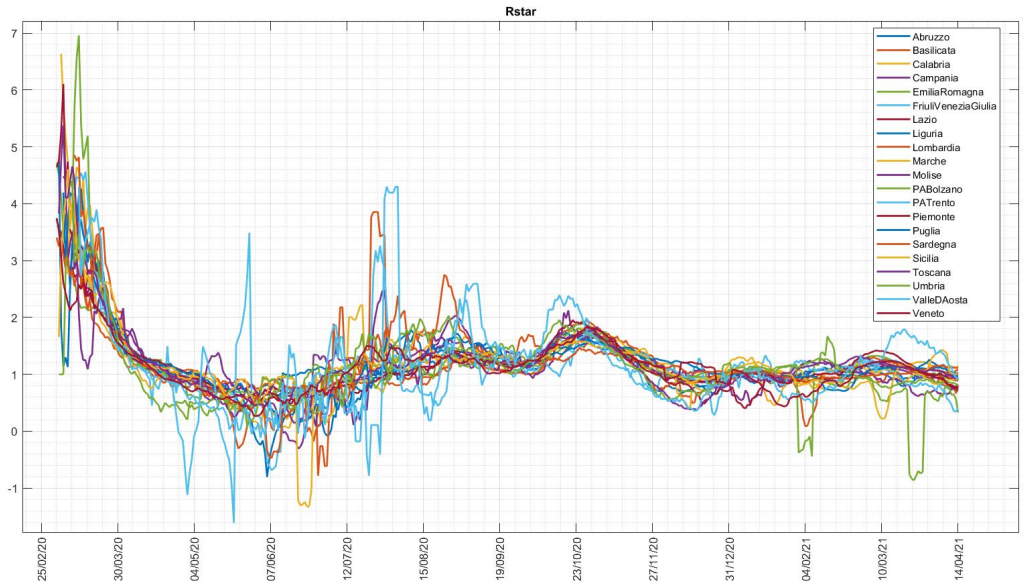
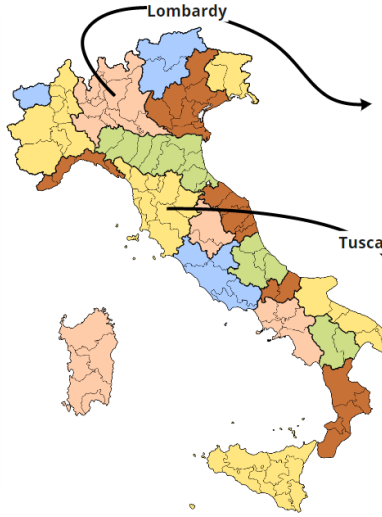


Figure 3: $\mathcal{R}^*$ for all regions for the whole period

3.3 The library: model-learning master

Model-learning master [Reg] is a machine learning Matlab library for model learning and model order reduction applications of Artificial Neural Networks. As underlined in Section 2, our interest lies in particular in the data-driven model-learning side of the library, due to the intrinsic nature of our task: finding the law standing underneath the Basic Reproduction Number of the ongoing pandemic, based on real world data.
For the complete documentation about the library we refer to [Reg], the following will serve as a brief explanation of its use and its role in the report.
The most important function in the library is `model_learn`: its purpose is to develop, train and test the Neural Network. It takes into account a problem option file with all the relevant information about the incoming tasks and uses all the options about the desired Neural Network features, chosen by the designer. In particular, it requires the directory of a training set where all the input-output

pairs, that are to be used for the optimization of the network's parameters, can be found. The training set must be organized as an array of cells, each containing a struct whose fields contain all the inputs and the associated measured output, as explained in Figure 4.



DAY 1	DAY 2	DAY 3	DAY 4	...
Hospitalized day 1	Hospitalized day 2	Hospitalized day 3	Hospitalized day 4	...
Severity of Restrictions day 1	Severity of Restrictions day 2	Severity of Restrictions day 3	Severity of Restrictions day 4	...
Deceased day 1	Deceased day 2	Deceased day 3	Deceased day 4	...
R* day 1	R* day 2	R* day 3	R* day 4	...

DAY 1	DAY 2	DAY 3	DAY 4	...
Hospitalized day 1	Hospitalized day 2	Hospitalized day 3	Hospitalized day 4	...
Severity of Restrictions day 1	Severity of Restrictions day 2	Severity of Restrictions day 3	Severity of Restrictions day 4	...
Deceased day 1	Deceased day 2	Deceased day 3	Deceased day 4	...
R* day 1	R* day 2	R* day 3	R* day 4	...

Figure 4: Two examples of Training Samples regarding the Regions of Lombardy and Tuscany.

For a detailed description of the main design and problem options to provide to model-learn see Subsection 3.4.1 where all the training choices are reported and commented. Our attempts at reaching a robust learning process of $\mathcal{R}^*$ with the usage of the model-learning master library can be summarized, in a few steps, as follows:

Scheme Of the Library Usage:

- obtain the data from the repository
- reorganize properly the data into a dataset with the correct structure
- define the problem specifics and choose the design options of the network in an opt_problem.ini file (opt_R_estimate in our case)
- run model_learn obtaining the trained Neural Network
- finally pass to the calibration of the SEIRD model (Section 4)

3.4 Training tests

These test were done with the aim of exploring all the possibilities in the definition of the Neural Network itself and the Input/Output pairs in order to be able to achieve the most efficient learning process and to obtain good results on a consistent basis. Our approach is that of *trial and error*. Our research was mainly concerned about finding the optimal values for the hyperparameters of the network, because even though they do not take part into the optimization process they are crucial for avoiding phenomena such as over or under fitting. The number of different options was very high and in exploring all the possibilities, we have followed the principle of *parsimony*. We always started with the fewer options and the simplest network.

3.4.1 Results of the training tests

In this section we report the main training tests that we have made, observing what we have tried and what we have learned. We divide in three main areas the different observations

1. Samples construction
   - *time-length of each sample*:
     We have tried to understand which should be the length of the samples of the dataset. We noticed that both too long and too short I/O pairs resulted in a non optimal learning. We think that in the the latter case the information is simply not enough, whereas in the former case the relevant information is hidden inside to much irrelevant data. Our choice has been to keep the samples' length around 30 days, we found that this was the range that provided the best results in a consistent way.
   - *gathered around a pandemic wave, or sparse in the whole period*:
     We noticed that there were two interesting periods where the signals were not nearly constant and, in order to feed the NN with a sufficiently exciting training set, we have always tried to include I/O pairs from the periods of the two waves.
   - *overlapping or non-overlapping time spans*:
     We tried to include in the training set overlapping periods of time in such a way to better cover the periods with the most amount of information. This choice slightly improved the Network's results.
2. Input signals
   - *choice of inputs*:
     The choice of the inputs of the system has been a crucial aspect. We initially started considering the number of infectious individuals but we then abandoned it for the problems highlighted before, regarding the reliability of the data. In the end the input signals that we have chosen to use for the system were the number of hospitalized individuals, the number of deceased and the severity of restrictions index. We have selected this small subset of data because they are the most reliable ones, being at the center of the concern during the emergency and less affected by monitoring issues. In addition, we did not include the number of intensive care patients since due to the close relation to the number of hospitalized individuals we believe that it was an unnecessary complication.
   - *normalization to have uniform regions*:
     The 21 Italian Regions present several differences, for instance the size, the number of inhabitants, the geography of the area and even population density. Therefore, it is not reasonable to assume that all the Regions will behave accordingly to the same model for the determination of $\mathcal{R}^*$, the pandemic will inevitably spread differently in each one of them. In order to reduce differences (especially the one related to the different sizes), we have normalized all the data for the associated population at the start of the pandemic and we have seen an improvement of the results.
3. Network design and training features
   - *number of training samples*:
     Especially in this section, where the training was done exploiting the data of either all or of a subset of the 21 Italian Regions, we explored the possibility of large training sets. We noticed that a richer training set usually results in a better learning, considering the high number of scenarios that the Network can learn from, but it is important not to provide to the algorithm an excessively large number of unexciting samples. The Network would then not capture the relevant traits and the training will not be satisfying. Furthermore, we noticed that with a greater training set, it became easier to mismatch the complexity of the NN and the complexity of the dataset and phenomena of overfitting and underfitting became harder to avoid. See Figures 5 and 6.

- *number of neurons*:
  The number of neurons, as mentioned above, must be determined accordingly to the number of training samples the Network has to learn from. It must not be too small nor to high. The risks are underfitting and overfitting, that are usually very clear to identify. We refer again to Figure 5 and Figure 6. We report later in the Table 1 the rule of thumb that we have followed: it associates a suggested number of neurons to a range of number of training samples. Note that this is not a true rule and the boundaries should be taken as approximate estimates since there is no sharp separation between the regions.
- *number of states*:
  We have seen that as the training set gets larger the Net must follow with its complexity, that can be augmented, as explained before, adding neurons or increasing the number of states of the learned model. We have preferred to keep this value as small as possible since with NN it is not possible to interpret what these states represent. For most of our project, the number of states has been kept equal to one, and since we have put ourselves in the *Input-inside-the-state* scenario it coincides with the $\mathcal{R}^*$
- `fixed_x0`:
  When it comes to learning from I/O pairs the library left two possibilities in the determination of the starting points of the learned curves. It it possible to set the hyperparameter `fixed_x0` $= 1$ (true) and the result is that the learned signals will all start from the average of all the starting points of the dataset solutions and will not change in the learning process. Otherwise one can set `fixed_x0` $= 0$ (false) and the Network' solution will match the starting point of each sample independently and it will eventually span in a fixed range during the optimization.
  We found out that the latter option was very interesting when the number of training samples was sufficiently high and the training set presented non homogeneous samples with very different starting points in their $\mathcal{R}^*$. In this case the mean value would completely miss some signals. In the end we preferred to keep the option true because of two reasons. The first one is that we wanted to have a test set with which to check the learning process and the option `fixed_x0` $= 0$ did not allow that. The other reason is that for the task of Model Calibration, the NN had to be solved on the set period of time and a NN trained with `fixed_x0` $= 0$ could not be applied on a new test. Nevertheless, this choice often provided a good fit and we were satisfied with the results.

Number of neurons	2	3	4	5	...
Training samples Range	2-10	6-30	21-60	55-110	...

Table 1: Number of Neurons in function of the size of the training set (number of I/O pairs)

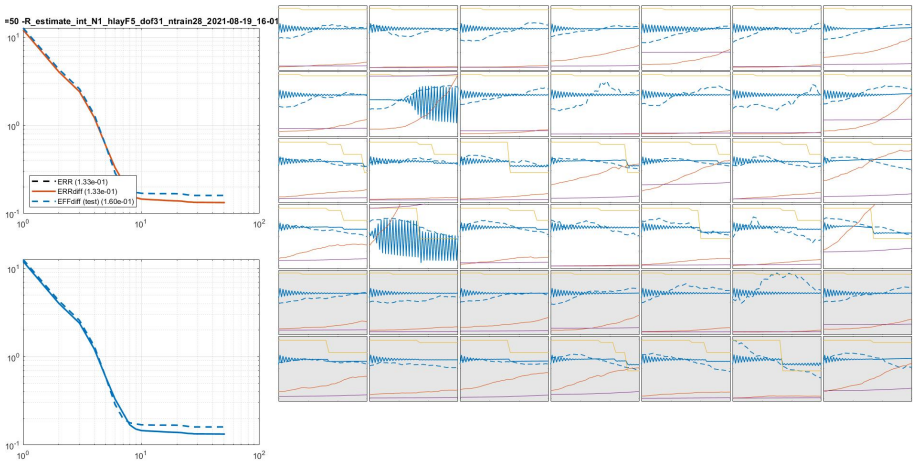


Figure 5: Example of overfitting

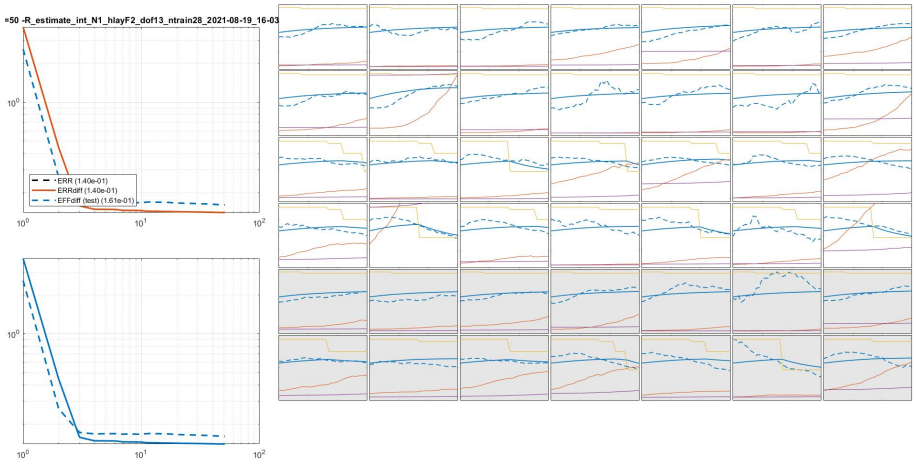


Figure 6: Example of underfitting

Remark. *Figure 5 is the result of a 5 neurons Neural Network's learning process based on 28 samples, 7 of the Italian Regions in two periods of 30 days centered around the second wave of the epidemic. Notice the characteristic oscillations and the high variance of the response of the Network.*
Figure 6 is instead the result of a 2 neurons Neural Network trained on the very same dataset. The variance of the responses is now very low and the different trends of the samples are not captured.
The figures also report the performance of the NN onto a test set. It is always important to check whether what the net has learnt can be applied to new samples and to understand if it can be truly generalized. The images show that the problems due to the non optimal training appear also in the test set and that the learning process has in this case failed.

3.4.2 Results

Until now we have performed several Training Tests in order to achieve a robust learning process and we have made observations on all the training hyperparameters' values that we deemed to be

the most relevant. In this paragraph we present and comment a few of the best training results that we have obtained. For instance, we show in Figure 7 and 8 two results in which the ANN presents a very good behaviour on the test set. The first is a 3-neurons ANN, whereas the second is a 4-neurons ANN, and they are both trained on a sampling of 33 I/O pairs of the periods of the first and second wave. The 9 test samples have been randomly selected from the very same period and the results, as it can be seen from the top left plot of the figure, are quite good since both the training and the test error decrease simultaneously. Moreover this example shows how the number of neurons does not really have a unique exact value.

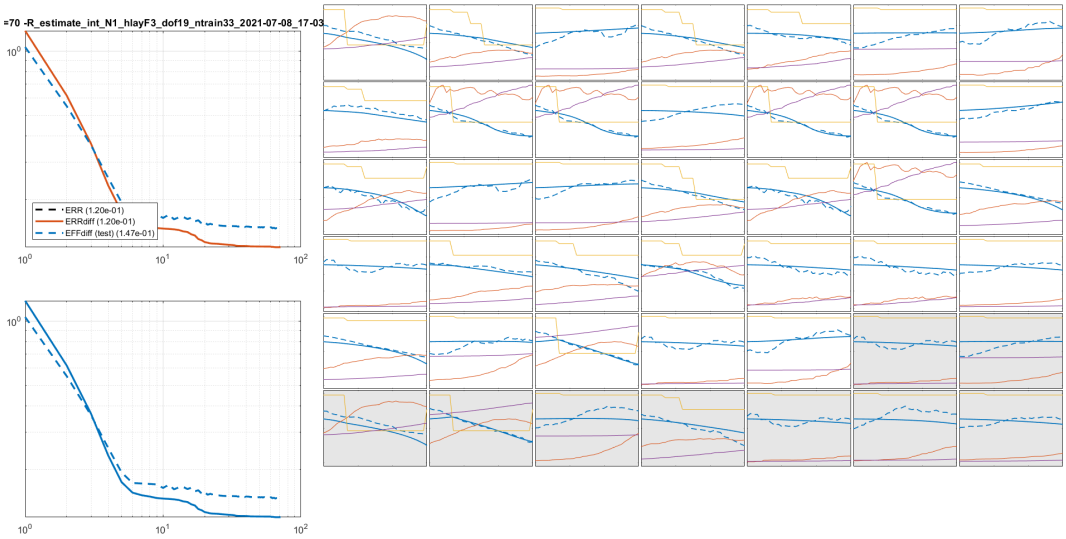


Figure 7: Training test 1

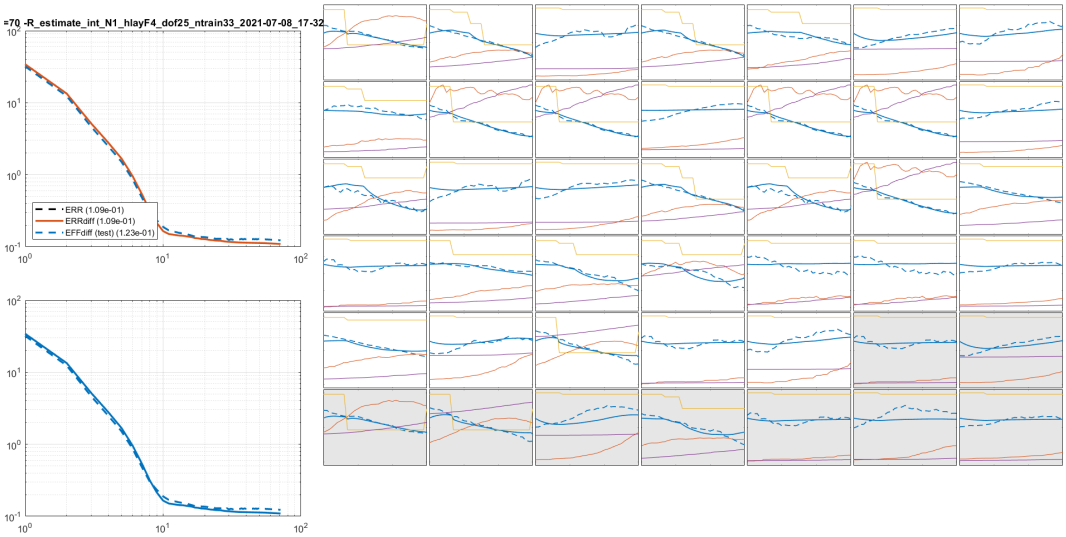


Figure 8: Training test 2

3.5 What-if analysis Tests

One of the main advantages of being able to infer the law standing underneath a quantity like the $\mathcal{R}^*$ is that it is possible to simulate different scenarios and compare the actual data with the model's one. This is fundamental to recognize and learn from past mistakes. We have taken inspiration from the online dashboard developed by the MOX Laboratory at Politecnico di Milano [MOX], which reports the estimated evolution of the disease, if harder restrictions had been applied earlier with respect to what has been done at the start of the second wave. We have tried to analyze with a trained Neural Network some of the choices that have been made in order to contain the pandemic. We have tried to explore the possibility of performing tests, in which a trained Neural Network is tested on some past periods of time with an arbitrary value of the severity of restrictions index, in order to analyze if different choices in the managing of the pandemic may have had different outcomes according to the NN.

3.6 Prediction Tests

Another significant application of an Artificial Neural Network is that of prediction. If the law has been identified properly the Network could be used to estimate the value of the quantity of interest in a future period, using projected inputs.
In our tests, gathered under the name of *Prediction Tests*, a NN trained on past data is given the predicted inputs for the future period of time and its solution provides the $\mathcal{R}^*$ predicted by the Network. The future inputs are obtained with regression models exploiting polynomials of different degree. We noticed that these tests are very fragile since there isn't a systematic way of choosing the polynomial degree: the latter must be set taking into consideration both the behaviour of the inputs in the selected training period of time and the length of the training and prediction time spans.
We report for example in Figure 9 the extrapolation of the input data on the number of hospitalized individuals in the 21 Regions. In this case a linear regression model has been used to extend the curves from the training to the prediction time span.

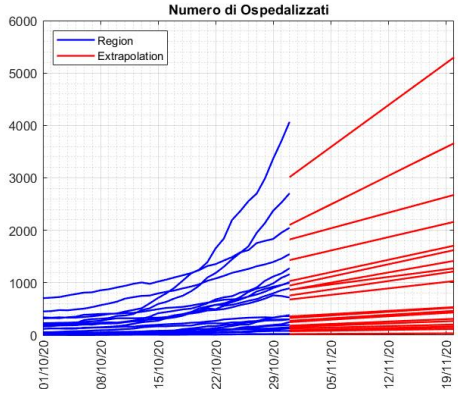


Figure 9: Prediction on the number of Hospitalized

4 Model Calibration

As previously explained in Section 2.2, a mathematical model for the description of real-world phenomena always relies on some parameters, whose values, in most cases, cannot be known *a priori* nor determined with precise formulas[5]. The model must be turned over and the parameter values must be obtained as the *implied* ones by virtue of minimizing the distance - in mathematical sense - between some known, measured, solution and the model's one.
Our project attempts at obtaining a better calibration of the parameters of epidemiological compartmental models, by determining in a data-based and parametric Machine Learning framework the initial condition for this optimization procedure. We have understood in Section 3 how to train an Artificial Neural Network in order to be able to obtain the Basic Reproduction Number from some input data. We now exploit our understandings to calibrate the model, starting from the value that our trained Neural Network will estimate for the period of time we are considering. The parameter that we are interested in is β, which represents the transmission rate of the disease and it is fundamentally linked to $\mathcal{R}^*$ thanks to this relation:

$$\beta = \gamma \mathcal{R}_0 \frac{N}{S} \tag{11}$$

Remark. *A fundamental difference with respect to Section 3 is that we will perform the calibration not focusing on the single regions, but on the data available in [Con] regarding Italy as a whole.*

Remark. *Calibration will be performed onto a discretization of the period going from 04/03/2020 to 31/05/2020 similarly to what is done in [Nic21]. Hence we will find a piece-wise constant in time (from here on, pwc) simplification of the true parameter β. See Table 2*

Period	I	II	III	IV	V	VI
from	04 Mar	09 Mar	19 Mar	29 Mar	18 Apr	03 May
to	08 Mar	18 Mar	28 Mar	17 Apr	02 May	31 May

Table 2: Time discretisation from 4th of March to 31st of May

Calibration Scheme:

The general functioning of the calibration procedure is the following:

(a) train a NN to learn $\mathcal{R}^*$ on real I/O pairs
(b) obtain the piece-wise constant version of the solution of the NN on the calibration period
(c) compute the piecewise constant β associated to the $\mathcal{R}^*$ obtained from the NN in the previous step thanks to formula (11)
(d) use the estimated β as the initial condition for the calibration procedure
(e) calibrate the model

[5] for diseases such as the flu, the contagion rate can, for example, be estimated quite accurately, since the flu has been studied for a long time with many "training samples" available. The same cannot be said, however, for COVID-19, whose characteristics were unknown for the majority of the first half of 2020.

4.1 SEIRD model

The SEIRD model (10) is one of the most famous epidemiological compartmental models used in mathematics to describe the evolution of a disease, spreading among a population. The model divides all the individuals in five different states each representing a status in which an individual could find themselves during the pandemic. Its popularity comes from the fact that it represents a well balanced trade off between model accuracy and usability.

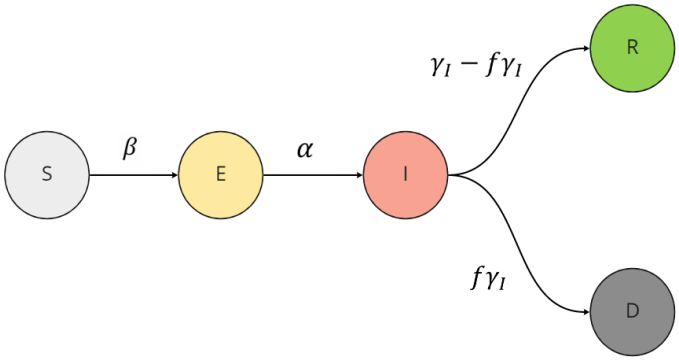


Figure 10: The SEIRD model scheme representation

In addition to the states of the SIR model (see (7)), two new compartments are introduced: D and E. The first stands for *Deceased* and allows to distinguish between an infectious individual that has recovered and a deceased one. The second one represents the *Exposed* individuals, that is individuals who have contracted the disease, but that are still in the incubation period.
The ODEs that characterize the SEIRD model are the following:

$$\begin{cases} \frac{dS}{dt} = -\frac{\beta IS}{N} \quad t \in (0,T] \\ \frac{dE}{dt} = \frac{\beta IS}{N} - \alpha E \quad t \in (0,T] \\ \frac{dI}{dt} = \alpha E - \gamma_I I \quad t \in (0,T] \\ \frac{dR}{dt} = (\gamma_I - f\gamma_I) I \quad t \in (0,T] \\ \frac{dD}{dt} = f\gamma_I I \quad t \in (0,T] \end{cases} \tag{12}$$

Notice that each ordinary differential equation of system 12 requires a prescribed initial condition. Moreover, each parameter has a specific physical meaning:

Parameters:	Meaning:
β	infection rate
α	average incubation rate
γ_I	average infectious time
f	infectious fatality rate
N	Italian population

The calibration of epidemiological models has one more layer of complexity since the choice of

the quantity on which to perform the tuning comes with several possibilities. Every state value would work, but the outcome will not be the same. One could find the parameters' implied values comparing the number of infectious individuals or even the number of exposed ones. The problem with these choices is that the errors and the inaccuracies on the chosen data will spread in the whole model through its parameters. This is why the decision must be done accurately, not to end up with a model whose precision has been based on shaky foundations. In our case, as already mentioned in the training phase, we think that the number of deceased individuals may be one of the most reliable datum among all the available ones, so the most reasonable choice should be to perform the calibration on this value. Furthermore, another justification of the choice of calibrating based on the number of deceased is that the number of COVID-19 deaths is the most concerning quantity in this situation of emergency, that needs to be accurately predicted by the model.
The procedure to find the optimal value for the piecewise constant parameter β, as mentioned before, is based on the minimization of the difference between the actual number of deceased and the one predicted by the model with the current parameter values. A simulation of the SEIRD model must start from an initial condition, namely the initial subdivision of the Italian population in the five compartments. Often the model is applied from the true start of the pandemic and the initial state is immediately set: the entire population in the susceptible state and just one individual is in the "Infected" compartment. In our case, due to lack of data, we had to start the calibration (and therefore also the simulation) from the 4th of March. The initial condition for the SEIRD model is in this case not easy to set. The problem arises in particular due to the state *Exposed* which cannot be found by definition, since it is the number of individuals that have already contracted the disease but are in the incubation period. We have introduced a parameter , namely `param`, which represents the ratio between the exposed individuals and the infected ones. In Section 4.3 it will be shown how we have tried to set it to different reasonable values following the available information and how we have selected one of them accordingly to the results provided.

4.2 Calibration algorithm

The following scheme provides a more detailed description of our approach in calibrating the SEIRD model using Artificial Neural Networks:

Calibration Algorithm

Preliminary steps:

- choose a suitable error norm
- choose a discretisation of the period 04/03/2020 - 31/05/2020
- choose the most objective[a] compartment of the SEIRD model

Calibration steps:

(a) train a NN to learn $\mathcal{R}^*$ on real I/O pairs
(b) obtain the piecewise constant version of the solution of the NN on the calibration period
(c) compute the piecewise constant β associated to the R obtained from the NN, thanks to formula (11)
(d) use the estimated β as initial condition for the calibration procedure
(e) solve the SEIRD model and compare the most objective compartment chosen with its real-world measurements. Tune the value of the parameter β to best resemble the data

[a] i.e. the one whose measurements are more reliable

In our code, we used a weighted H_1 norm[6] and considered the Deceased as the most objective compartment. Moreover, we divided the period under study in the six intervals described in Table 2.
In order to tune the value of the parameter β to best resemble the data, as explained in point (e), we used a sequential multistep optimisation procedure: we computed the initial guess β_0 either using ANNs or randomly (see Section 5.2 for a more detailed description) and fed this starting value to the function `fminsearch`, seeking for a new and better initial guess. We then used the function `fmincon`, which employed as starting point the previous output of `fminsearch`. Once the best β had been selected, the SEIRD model was solved using as contagion rate β itself and the computed state D was compared to the measured number of deceased individuals. We employed both functions `fminsearch` and `fmincon` in this specific order, since we found that the predicted value of D was particularly dependent on the initial guess of the minimisation procedure.

4.3 Calibration results

We report in Figure 11 the solution of a well-trained ANN onto the calibration period sample (from 04/03 to 31/05). It is possible to notice that the learned $\mathcal{R}^*$ is very close to the actual one, the only major difference is in the starting point, but that is due to the fact that the learned basic reproduction number must start from the average value of the $\mathcal{R}^*$ taken from the different samples of the training set. We also plot in Figure 11 the piecewise constant version of the learned curve, obtained computing the mean values on each subinterval of the discretization of the calibration period. We report below the evolution of the Deceased state predicted by the SEIRD model with the parameter β calibrated exploiting the machine learning starting point (β_{NN}, red line) and the one using the parameter obtained from a initial condition informed with

[6] in particular the squared norm is: $w_1\|D - \widehat{D}\|_2^2 + w_2\|D' - \widehat{D}'\|_2^2$, where $\widehat{D}$ is the estimated number of Deceased and w_1, w_2 are the weights

the data (β_{OG}, blue line). From a comparison with the registered number of deaths, the black dashed line, it is possible to assert that both the calibration techniques reach satisfying results, even though the one coming from the empirical R is definitely closer to the actual Deceased curve, especially in the first period. In Table 3 we explicitly collected the starting points and the calibrated parameters of the two different procedures in the example of Figure 12.

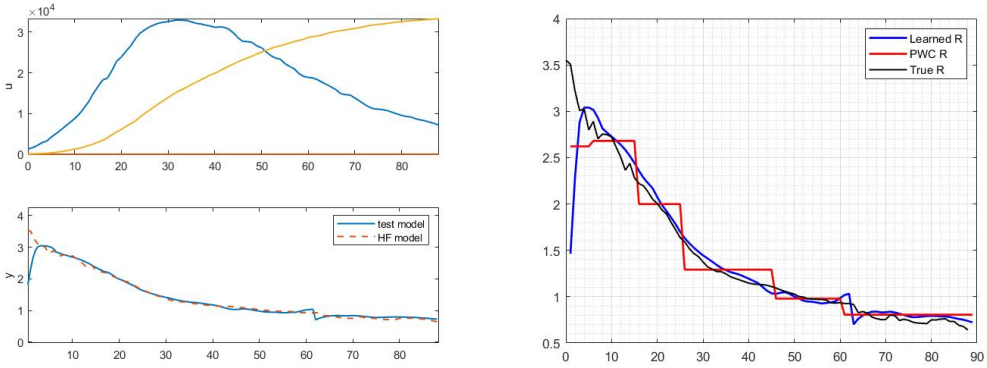


Figure 11: On the top left the inputs of the calibration period, on the bottom left the solution of the NN compared with the real one. On the right the pwc approximation (red) of the learned $\mathcal{R}^*$ (blue) that can be compared with the true $\mathcal{R}^*$ (black)

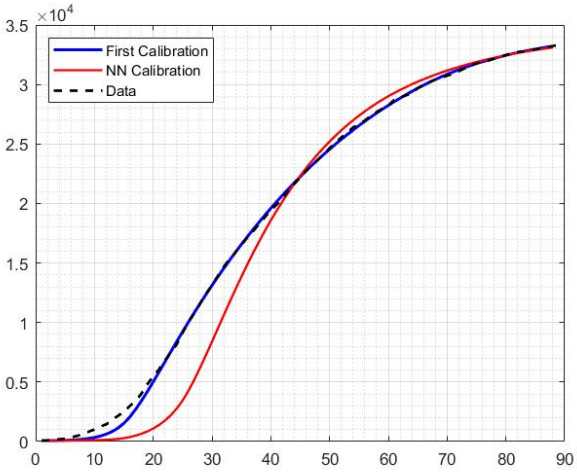


Figure 12: Calibration Result 1. In red the deceased curve obtained with β_{NN} and in blue the one with β_{OG}, the parameter calibrated starting from an empirical measure of beta

Starting point	$\beta_0(1)$	$\beta_0(2)$	$\beta_0(3)$	$\beta_0(4)$	$\beta_0(5)$	$\beta_0(6)$
First Calibration	0.2707	0.0913	0.0305	0.0192	0.0168	0.0388
ANN Calibration	0.1676	0.1713	0.1279	0.0827	0.0628	0.0517
Solution	$\beta(1)$	$\beta(2)$	$\beta(3)$	$\beta(4)$	$\beta(5)$	$\beta(6)$
First Calibration	3.7566	1.0260	0.0107	0.0335	0.0279	0.0044
ANN Calibration	0.4097	1.4078	0.5850	0.0017	0.0036	0.0123

Table 3: Starting Points and solutions of the two calibrations

We note that the result of the calibration is not the same for every run of the same algorithm. Indeed the optimization routine does not always reach the very same minimum of the cost functional and moreover everything is related to the Neural Network that we are using: different NNs provide different starting points for the calibration, since the learning process is not always the same. To stress this point we show in Figure 13 a calibration result obtained by a NN with the exact same structure of the previous one and with the same learning options but coming from a different run of `model_learn`.

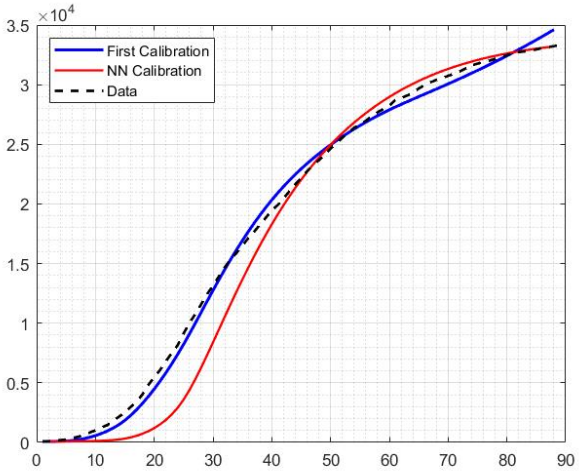


Figure 13: Calibration Result 2.

[We notice that in this specific calibration result the curve of the deceased predicted by the SEIRD model calibrated exploiting the ANN better depicts the last part of the actual curve with respect to the other calibration]

4.3.1 Conditions for the initial condition

As we have mentioned in Section 4.1, the task of calibrating the model comes with the problem of setting a proper initial condition for the number of exposed individuals on the 4th of March 2020. As highlighted above we have introduced a parameter, namely `param`, which represents the ratio between the number of infectious individuals and the number of exposed ones. We have initially set this value at 0.25 and we have later checked the sensitivity of our calibration procedure to the value of this parameter. We have ranged the values in the interval $[0.10, 0.70]$ and we have compared the solutions. The results, obtained by a well-trained Neural Network, are plotted

in Figure 14 and reported in Table 4 and show that the first calibration is not really affected by the value of `param` whereas there are two groups of solutions for the calibration procedure with ANN. The conclusion is that the correct value of `param` should be higher with respect to what we have initially set, since the distance from the Deceased curve is smaller in the case of `param` $\in [0.50, 0.70]$.

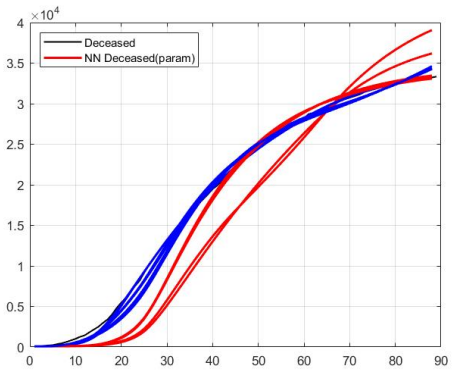


Figure 14: Sensitivity test for the value of `param`

Comment:
It is important to notice that the underestimation of the number of the deceased in the initial part of the plot (12) is a common feature of our results. We think that it is partially due to the fact that the NN must start from the average value of the $\mathcal{R}^*$ of the samples in the training set, since we must set `fixed_X0` to true (see Section 3.4.1). Since the initial value of the real $\mathcal{R}^*$ in the calibration period is usually higher than the average, the Neural Network needs time to adjust and it cannot be close to the real curve from the beginning. This initial lower $\mathcal{R}^*$ translates into a delayed and slower increase of the number of deceased.

	β_1	β_2	β_3	β_4	β_5	β_6
param = 0.10	0.4864	0.9818	0.7221	0.0440	0.0475	0.0043
param = 0.20	0.7561	0.9034	0.6854	0.0168	0.0879	0.0125
param = 0.30	0.4783	1.3500	0.5832	0.0014	0.0048	0.0091
param = 0.40	0.4897	1.3374	0.5681	0.0034	0.0009	0.0168
param = 0.50	0.5360	1.2921	0.5509	0.0069	0.0082	0.0034
param = 0.60	0.5333	1.3016	0.5340	0.0060	0.0080	0.0004
param = 0.70	0.5869	1.2550	0.5259	0.0071	0.0111	0.0033

Table 4: Results from the calibration robustness test with the ANN. β_i represents the value of the piecewise constant approximation of $\beta(t)$ in the i-th subinterval of the discretization of the period from the 4th of March to the 31st of May. See Table 2.

4.3.2 Comparison between the calibration methods

Our code needs to offer the "best" initial guess of β (based on the estimation of $\mathcal{R}^*$, provided by the Neural Network) to the minimisation function, namely `fminsearch`, whose job is to find the best value of the parameter under study. An alternative to NN, could be to choose a random

initial guess: in this last section of our project we wanted to test whether the use of the estimation of the basic reproduction number would yield better results, when compared with a random choice for the initial value of parameter β[7].
We found that, as intuition would suggest, the more the estimation of the Basic Reproduction Number resembles the true $\mathcal{R}^*$ shown in black in Figure 11 on the right, the better the minimisation performed by `fminsearch`, when fed by the initial guess computed with ANNs.
The comparison was composed of two stages:

- **quantitative comparison:** we deemed our algorithm better than the random choice for the initial value, if

$$w_1\|D - D_{NN}\|_2^2 + w_2\|D' - D'_{NN}\|_2^2 \leq w_1\|D - D_{IG_i}\|_2^2 + w_2\|D' - D'_{IG_i}\|_2^2 \quad \forall\, i = 1, ..., N_{trials} \tag{13}$$

  where N_{trials} is the number of trials, w_1 and w_2 are suitable weights, D is the measured number of deceased people, D_{NN} is the number of deceased individuals computed with the SEIRD model with β *obtained via ANNs* and D_{IG_i} is the number of deceased computed with the SEIRD model with β obtained with the i-th *random choice* for its initial value in the minimisation procedure.
- **qualitative or graphical comparison:** we compared the two algorithms by observing which one allowed to compute the parameter β, such that the state D of the SEIRD model would best resemble the measured curve.

In the plot 15, the fifteen blue lines represent the value of the Deceased computed with a random initial guess for β, while the red curve is the same quantity, when the initial guess is informed by our Neural Network. The dashed black line represents the measured number of Deceased.

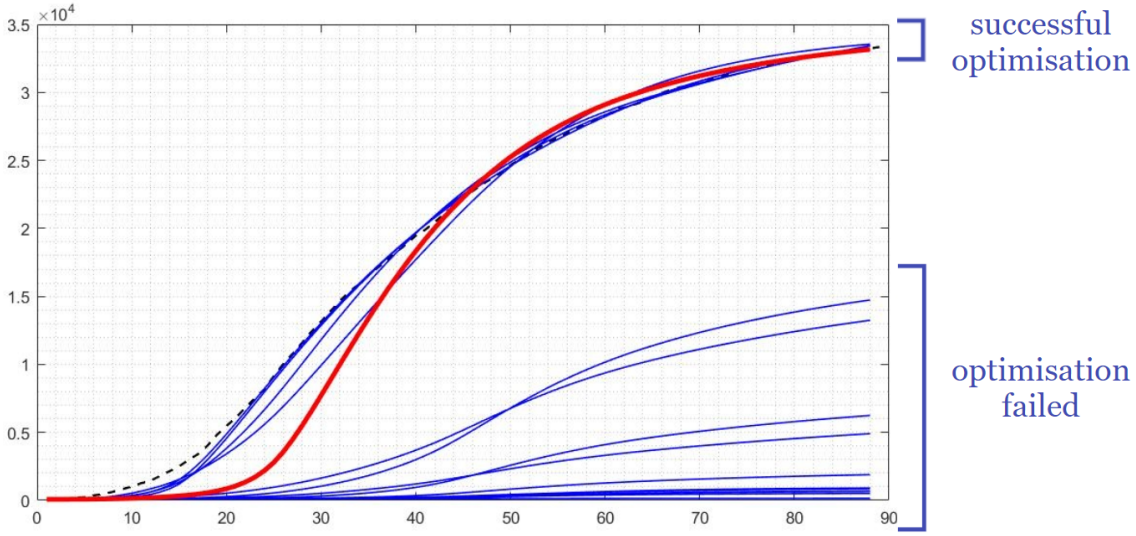


Figure 15: Performance comparison. Some blue curves are clearly the result of a failed optimization. This underlines the importance of properly setting the starting point of the minimisation.

We conclude that whenever real-world data cannot be used to make a guess for the initial value of β and the only information available is a rough interval for the components of the

[7] A scenario that could be realistic in the case in which it is unfeasible to obtain a reliable empirical measure of β

parameter vector, a trained NN might be a useful alternative.
Quantitatively, as shown Figure 16, a NN-informed initial guess achieves (for most of the trials) a lower H_1 norm, as defined in (13):

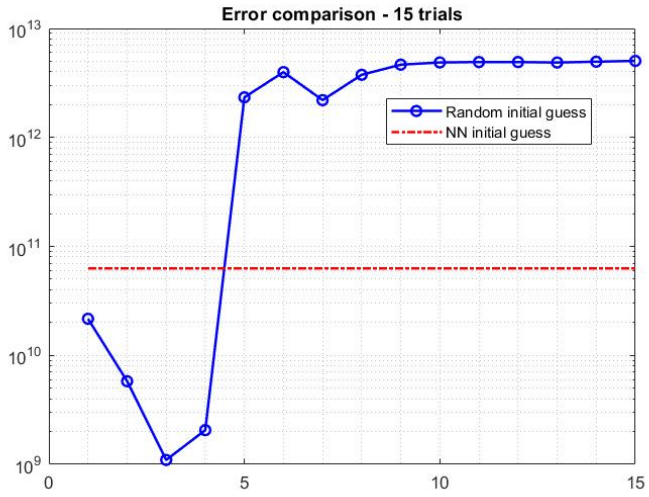


Figure 16: weighted H_1 norm of the error vs number of trial

5 Conclusion

Exploiting the applications of Artificial Neural Networks to data-driven model identification with the goal of obtaining a refined estimation of the transmission rate parameter in the SEIRD model, we have shown an alternative calibration technique that could be applied when it is not possible to exploit a data-informed starting point. The new calibration procedure requires only a sufficiently large and well diversified training set and the experience of the designer in the choice of the hyperparameters for the learning process. Moreover, another relevant aspect is that with a well-trained ANN it is possible to perform several calibrations, on different periods of time, without having to obtain an estimated measure of the wanted parameter on each calibration period, but just collecting the curves of the necessary inputs. This could turn out to be useful in a situation in which it is not feasible or relatively expensive to obtain measures of the unknown parameter.

On the other hand the main limitation of ANNs is the fact that, because of their black-box and data-based nature, the quality of the training samples needs to be sufficiently high in order to obtain a satisfying training process. This is also true for our algorithm, if fed with samples completely inconsistent with the calibration period the output of the trained ANN would diverge from the true one. In addition, another crucial aspect is the reliability of the available data. ANN are very good at recognizing small inconsistencies but major flaws in the dataset could ruin the whole procedure.

We note that the new calibration technique could be generalized even to more complex models, like SUIHTER, where the number of parameter to tune correctly is higher and the problem of calibration even more significant.

Furthermore, it could be possible to consider an integrated model where the ANN is included in the epidemiological compartmental one. For instance an ANN-SEIRD model should be driven by six ordinary differential equations, five coming from the usual SEIRD and one being the learned differential law followed by one of the parameters. Such a model could be exploited in a combined calibration procedure to find the implied value of one of the inputs of the ANN, for instance the implied value for the severity of restrictions index. This could provide insights into the real impact of the restrictions imposed by the government.

References

[Fra19] Francesco Regazzoni, Luca Dedè, Alfio M. Quarteroni. "Machine learning for fast and reliable solutions of time-dependent differential equations". In: *Journal of Computational Physics* 397.108852, (2019). DOI: https://doi.org/10.1016/j.jcp.2019.07.050.

[Nic21] Nicola Parolini, Luca Dedè, Paola F. Antonietti, Giovanni Ardenghi, Andrea Manzoni, Edie Miglio, Andrea Pugliese, Marco Verani and Alfio M. Quarteroni. "SUIHTER: A new mathematical model for COVID-19. Application to the analysis of the second pandemic outbreak in Italy". In: (2021).

[Bat] Roberto Battiston. *Un modo semplice per calcolare R(t)*. URL: https://www.scienzainrete.it/articolo/modo-semplice-calcolare-rt/roberto-battiston/2020-11-20.

[Con] Presidenza del Consiglio dei Ministri - Dipartimento della Protezione Civile. *COVID-19 Italia - Monitoraggio situazione*. URL: https://github.com/pcm-dpc/COVID-19.git.

[MOX] Politecnico di Milano MOX Department of Mathematics. *epiMOX Dashboard*. URL: https://www.epimox.polimi.it/.

[Reg] Francesco Regazzoni. *Model Learning library*. URL: https://github.com/FrancescoRegazzoni/model-learning.

[San] ISS - Istituto Superiore di Sanità. URL: https://www.iss.it/.

[Sta] Istat - Istituto Nazionale di Statistica. URL: http://dati.istat.it/Index.aspx.

Appendix

A Data plots

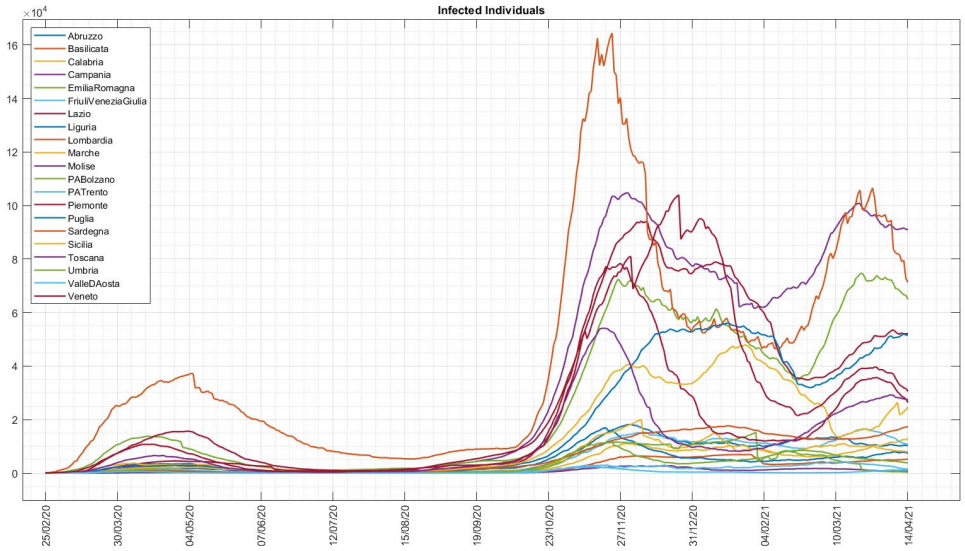


Figure 17: Number of infected individuals

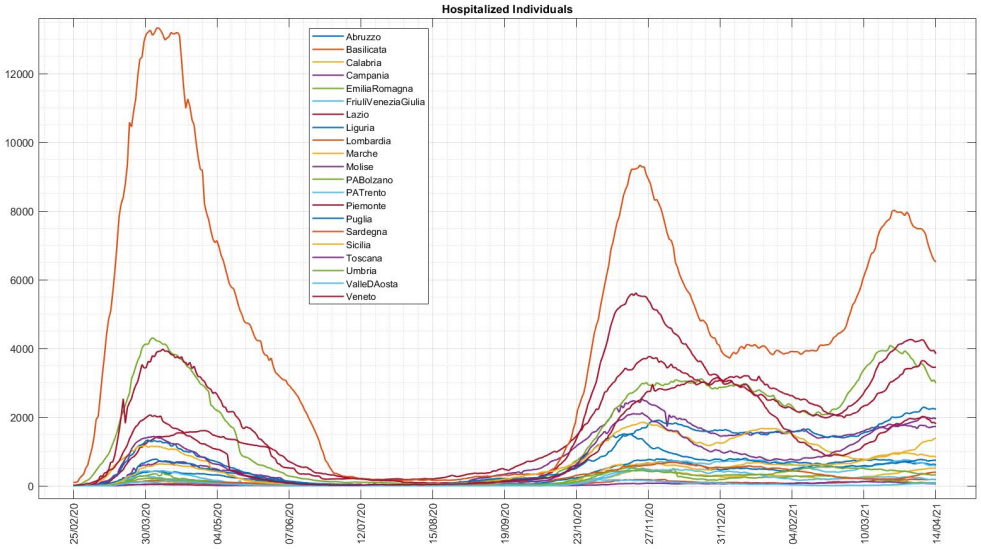


Figure 18: Number of hospitalized individuals

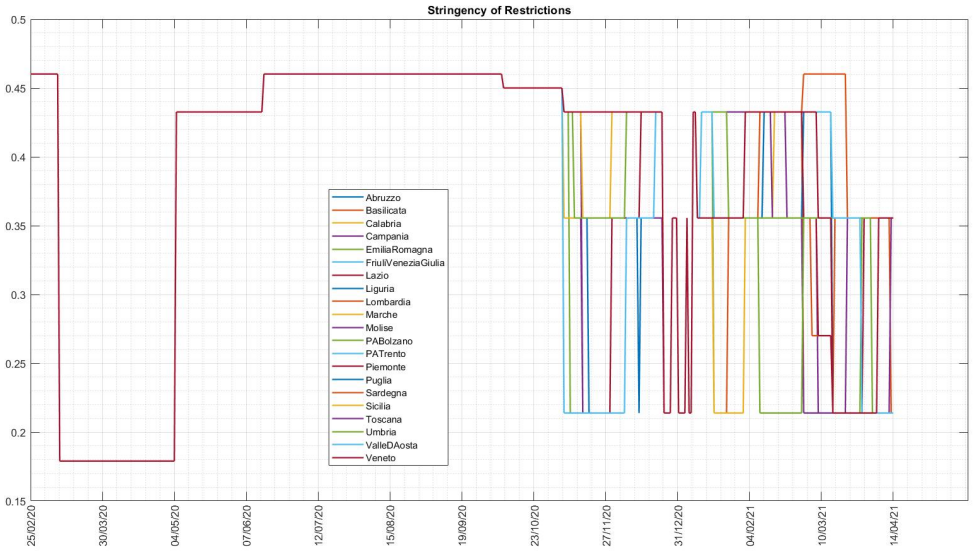


Figure 19: Severity of restrictions index

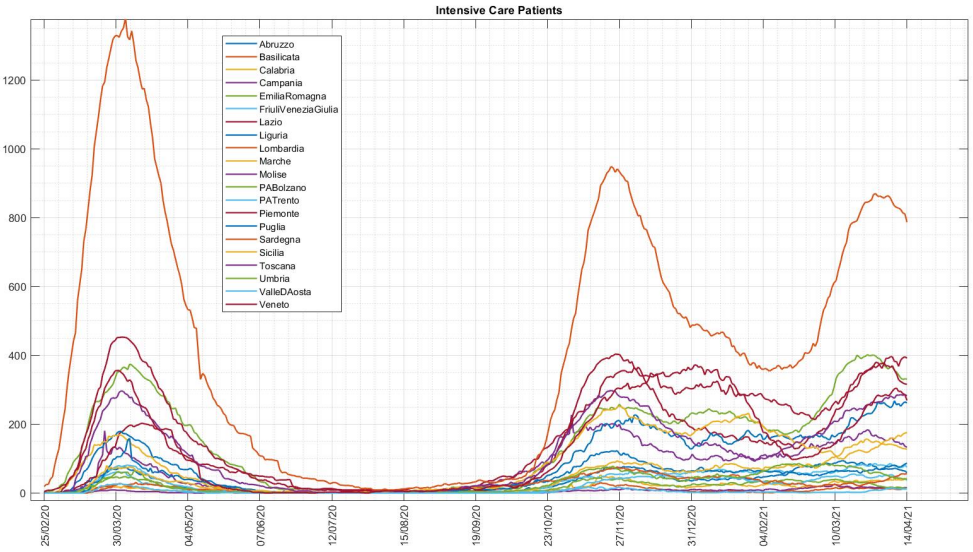


Figure 20: Number of intensive care units patients

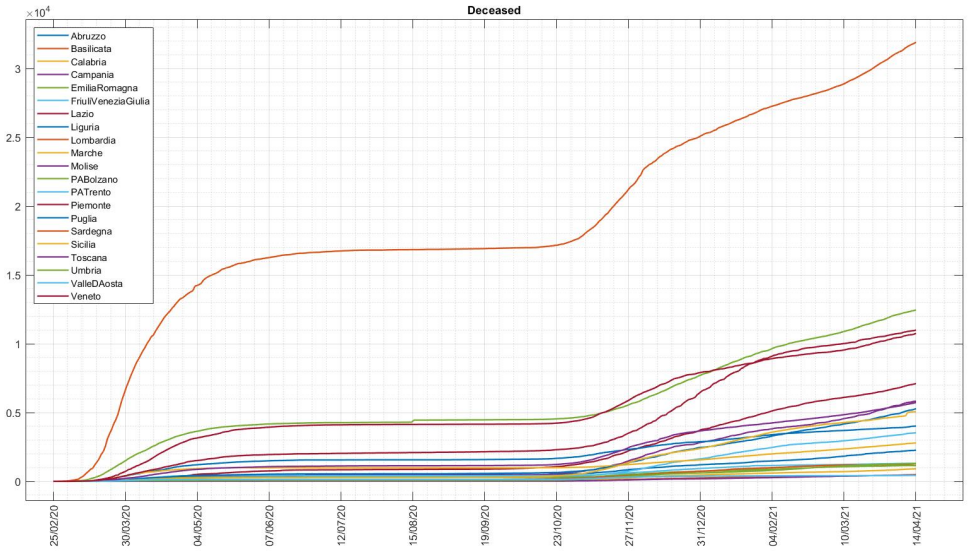


Figure 21: Number of deceased individuals

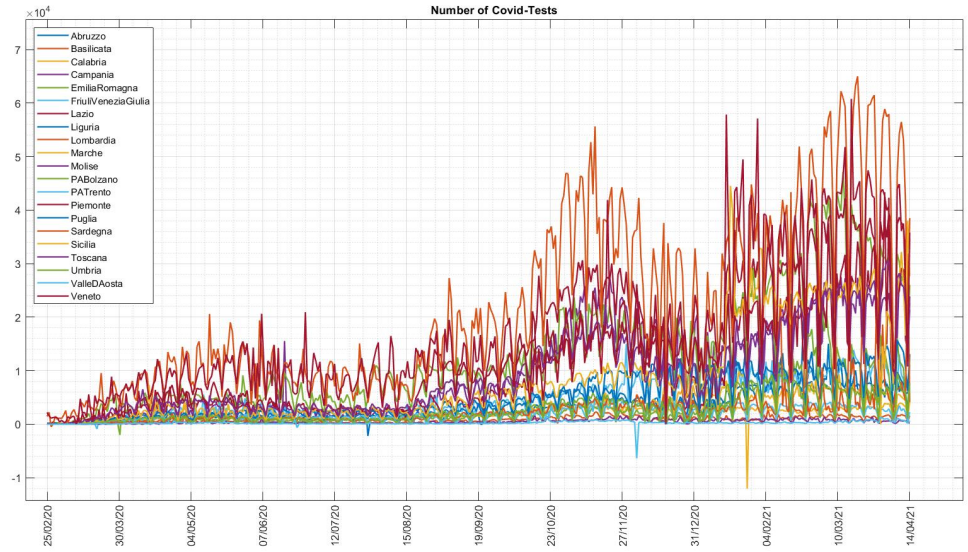


Figure 22: Number of COVID-tests performed

B Our code

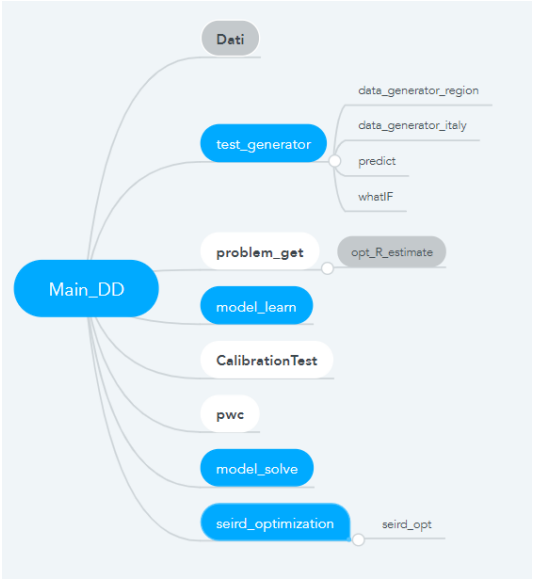


Figure 23: Structure of the code

The function `mainDD.m` is the one at the core of our work and the structure of the code is reported in Figure (23). We have highlighted in blue the main passages and in grey the functions where the designing options are specified.

`main_DD.m` takes the data for the test that it has to run from a function `Dati.m` that we have organized as a repository where it is possible both to find some of the tests we have used as examples and all the options to define a new one. We have defined a switch to pass from one purpose of usage of the NN to another one, altogether there are four of them and each, despite being all defined in the very same function `Dati.m`, is then treated differently.

switch	Usage
t	training of the neural network
a	what if analysis
p	prediction analysis
c	calibration of model

After having obtained the relevant data and parameters, `main_DD.m` calls `test_generator.m` which, accordingly to the chosen switch, the training set (`Tests_training`) and the test set (`Tests_testing`) are defined from the data available in [Con]. At this stage it is necessary to report in the opt file the options that appear on display since they are necessary to define the problem specifics and to design the Neural Network.
Then, `model_learn.m` builds and trains the Artificial Neural Network.
Finally, the trained NN is solved by `model_solve.m` onto the calibration period, assembled by `CalibrationTest.m`. The pwc version of the learned $\mathcal{R}^*$ is computed with `pwc.m`. Finally the calibration is performed by `seird_optimization.m`.