Final Project

Improving Time-Series Forecasting for the COVID-19 Pandemic with Invertible Neural Networks and Gaussian Processes: A Comparison

Leonard Marks, Maximilian Richter

`https://github.com/maxawake/INN-COVID19`

Advanced Machine Learning

Universität Heidelberg

Summer Semester 2021

Contents

1 Introduction

by Maximilian Richter

Since the beginning of the COVID-19 pandemic in December 2019, the interest in improving epidemiological and statistical models has grown in an unprecedented way. It is of global interest to make better estimations and predictions of the future, in order for politics and society to decide which counter measures might be useful to successfully contain the viral spread of a potentially deadly disease. Especially promising for such tasks are methods from the area of machine learning. Not only can machine learning models handle sparse and noisy data when regressing models to real world data, but the statistical framework in which they usually are formulated make them perfect candidates for assessing possible underlying dynamics.

In this project we want to evaluate this strength of machine learning by comparing two rather different state-of-the-art models which have already been successfully applied to real-world datasets of the COVID-19 pandemic . The models under investigation are the BayesFlow network [8], which involves an Invertible Neural Network (INN), and Gaussian Process Regression (GPR) [4]. Even though Gaussian Processes are, by its nature, parametric-free models and BayesFlows are simulation-based parametric models, both are formulated in the Bayesian framework of probability theory and statistics. It is this feature that make both models appealing for comparison, since both are able to yield uncertainty estimations for their respective predictions.

Following this we will be taking a closer look into the classical compartment simulation of infectious diseases, namely the SIR-model and its advancements building upon. It is not only important to understand the dynamics of such a system in order to interpret the results, but also does the BayesFlow depend on the simulation as training samples. How the BayesFlow and Gaussian Processes Regression work is briefly discussed in section 3 and 4, respectively. The results of our analysis are portrayed in section 5. As a conclusion we will discuss our results in section 6.

2 Modeling Infectious Diseases

by Leonard Marks

The Understanding of how an infectious disease spreads through a chain of infections is well known for most diseases (COVID-19, HIV, etc.), but the dynamics in a population is without mathematical methods a highly complex task.

Three different basic types of deterministic models can help to understand the dynamic of an infectious disease better and lead to more complex models as further knowledge is acquired by new insights [6]. (i) The **SIS**-model is divided into two classes, the susceptible-class (**S**) and the infected-class (**I**). The model describes that any susceptible can become infected and return to be susceptible, this excludes the possibility of immunity and so does not con cure with our goal-model. (ii) The **SIR**-model without vital dynamics is divided into one more class than the first model. The removed-class (**R**) implies the deaths from the disease and people who were infected but recovered (with immunity). This model corresponds mostly to the behaviour and basic dynamics of COVID-19. (iii) The **SIR**-model with vital dynamics is almost the same as the second model, but includes the natural birth rate and death rate over decades and as such is not yet of interest for the COVID-19 modelling.

2.1 SIR Model

The SIR model is one of the most basic models for describing the temporal dynamics of an infectious disease in a population [2]. For a time step t the differential equations for our 3 different classes susceptible S, infected I and removed R are given by

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\frac{\beta IS}{N} \tag{1}$$

$$\frac{\mathrm{d}I}{\mathrm{d}t} = \frac{\beta IS}{N} - \gamma I \tag{2}$$

$$\frac{\mathrm{d}R}{\mathrm{d}t} = \gamma I \tag{3}$$

with their corresponding parameters in Table 1 and N being a constant which is the sum of all classes over the observation period.

$$S + I + R = N \tag{4}$$

This does give some insight into the behaviour of the basic behaviours of infectious diseases, but to be implied for a certain one like COVID-19, it does not suffice in its complexity and should consider more detailed classes and further connections between these.

2.2 Advanced SEIR-Type Models

One example for a more complex model would be the SEIR-type model [8], which expands the basic SIR-model by E - infected individuals who do not show symptoms and are not yet infectious, C - infectious individuals who recover without being detected, I - symptomatic cases that are infectious and split R into R - recovered and D - dead. The true time-series needs to be estimated as they are considered latent. Therefore our ordinary differential equations are given by:

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\lambda(t)\left(\frac{C+\beta I}{N}\right)S \tag{5}$$

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \lambda(t)\left(\frac{C+\beta I}{N}\right)S - \gamma E \tag{6}$$

$$\frac{\mathrm{d}C}{\mathrm{d}t} = \gamma E - (1-\alpha)\eta C - \alpha\theta C \tag{7}$$

$$\frac{\mathrm{d}I}{\mathrm{d}t} = (1-\alpha)\eta C - (1-\delta)\mu I - \delta d I \tag{8}$$

$$\frac{\mathrm{d}R}{\mathrm{d}t} = \alpha\theta C + (1-\delta)\mu I \tag{9}$$

$$\frac{\mathrm{d}D}{\mathrm{d}t} = \delta d I \tag{10}$$

parameters and priors are defined by Table 1.

Since the beginning of the COVID-19 outbreak and the increasing numbers of infected people, countries had to intervene with IPC (Infection Prevention and Control) measures to prevent rapid exponential growth of infected people, these measures are taken into account by the factor $\lambda(t)$ (Table 2). This time-varying transmission rate reduces the

Parameter	Symbol	Prior Distribution
Number of initially exposed individuals	E_0	$\text{Gamma}(2, 30)$
Risk of infection from symptomatic patients	β	$\text{LogNormal}(\log(0.25), 0.3)$
Rate at which exposed cases become infectious	γ	$\text{LogNormal}(\log(1/6.5), 0.5)$
Rate at which symptoms manifest	η	$\text{LogNormal}(\log(1/3.2), 0.5)$
Rate at which symptomatic individuals recover	μ	$\text{LogNormal}(\log(1/8), 0.2)$
Rate at which undiagnosed individuals recover	θ	$\text{Uniform}(1/14, 1/3)$
Rate at which critical cases die	d	$\text{Uniform}(1/14, 1/3)$
Probability of remaining undetected/undiagnosed	α	$\text{Uniform}(0.005, 0.99)$
Probability of dying from the disease	δ	$\text{Uniform}(0.01, 0.3)$

Table 1: Description of disease model parameters and corresponding prior distributions [8]

transmission rate at three time points, each time point is represented by a piece-wise linear function with the effect of strength, time interval for the effect and the effect to fully manifest itself as their degrees of freedom.

Since our data-set is the RKI (Robert-Koch Institute) collected information of infected cases and deaths by day, the error source of delayed reports, different reporting activities over the week as well as random fluctuations needs to be put into consideration. Only three classes of the SEIR-model are observable and are given by

$$I_t^{(\text{obs})} = I_{t-1}^{(\text{obs})} + (1 + f_I(t))(1-\alpha)\eta C_{t-D_I} + \sqrt{I_{t-1}^{(\text{obs})}}\sigma_I \xi_t \tag{11}$$

$$R_t^{(\text{obs})} = R_{t-1}^{(\text{obs})} + (1 + f_R(t))(1-\delta)\mu I_{t-D_R} + \sqrt{R_{t-1}^{(\text{obs})}}\sigma_R \xi_t \tag{12}$$

$$D_t^{(\text{obs})} = D_{t-1}^{(\text{obs})} + (1 + f_D(t))\delta d I_{t-D_D} + \sqrt{D_{t-1}^{(\text{obs})}}\sigma_D \xi_t \tag{13}$$

with f_I, f_R, f_D being the weekly modulation scalars which are computed with the priors (Table 3) as followed

$$f_C(t) = (1 - A_C)\left(1 - \left|\sin\left(\frac{\pi}{7}t - 0.5\Phi_C\right)\right|\right), \qquad C \in \{I, R, D\} \tag{14}$$

Parameter	Symbol	Prior Distribution
Onset date of each change to take effect	t_1	Normal(2020/03/09) = Day 8,3)
	t_2	Normal(2020/03/16 = Day 15,3)
	t_3	Normal(2020/03/23 = Day 22,3)
	t_4	Normal(2020/05/06 = Day 66,3)
Duration of each change to fully manifest itself	Δt_j	$\text{LogNormal}(\log(3), 0.3)$
Transmission rates before / after each change	λ_0	$\text{LogNormal}(\log(1.2), 0.5)$
	λ_1	$\text{LogNormal}(\log(0.6), 0.5)$
	λ_2	$\text{LogNormal}(\log(0.3), 0.5)$
	λ_3	$\text{LogNormal}(\log(0.1), 0.5)$
	λ_4	$\text{LogNormal}(\log(0.15), 0.5)$

Table 2: Description of intervention model parameters controlling the time-varying transmission rate [8]

All these, the disease model, intervention model and observation model, conclude our epidemic model.

Parameter	Symbol	Prior Distribution
Reporting delays	$D_{C\in\{I,R,D\}}$	$\text{LogNormal}(\log(8), 0.2)$
Weekly modulation amplitudes	$A_{C\in\{I,R,D\}}$	$\text{Beta}(0.7, 0.17)$
Weekly modulation phases	$\Phi_{C\in\{I,R,D\}}$	$\text{vonMises}(0.01)$
Reporting noise scale	$\sigma_{C\in\{I,R,D\}}$	$\text{Gamma}(1, 5)$

Table 3: Description of observation model parameters controlling reporting properties [8]

3 BayesFlow

by Maximilian Richter

In the past, artificial neural networks have proven to be a versatile and robust machine learning method, far beyond scientific purpose only. For science in particular, Bayesian neural networks are extraordinary well suited, since they are formulated already in a statistical framework.

Given a configuration of hidden parameters $\boldsymbol{\theta}$, the forward process is finding a suitable model of the mechanism that generates observations. The inverse process, on the other hand, is required to infer the hidden states of a system from measurements. Since crucial information is lost in the forward process, the inverse process is often intractable and ill-posed (e.g. several parameter combination yield the same observed data) [3].

A rather novel but very successful method for solving such inverse problems is the BayesFlow [7]. This method uses simulation to learn a global estimator for the mapping from observed data to underlying model parameters with invertible neural networks (INN) implemented as a normalizing flow.

3.1 Invertible Neural Networks

In order to counteract the inherent information loss of the forward process, invertible neural networks use additional latent output variables $\mathbf{z}$, which include information about $\mathbf{x}$ that it not contained in $\mathbf{y}$. The INNs learns to associate hidden parameter values $\mathbf{x}$

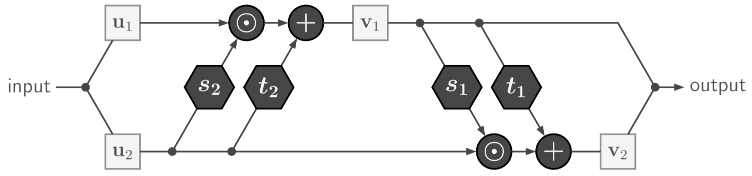

Figure 1: forward propagation [1]

with unique pairs $(\mathbf{y}, \mathbf{z})$ of measurements and latent variables. The forward training thus optimizes the mapping $(\mathbf{y}, \mathbf{z}) = f(\mathbf{x})$ and implicitly determines the inverse $\mathbf{x} = f^{-1}(\mathbf{y}, \mathbf{z}) = g(\mathbf{y}, \mathbf{z})$. Further, the density $p(\mathbf{z})$ of the latent variables is restricted to the shape of a Gaussian distribution. Therefore, the INN represents the posterior distribution $p(\mathbf{x} \mid \mathbf{y})$ by a deterministic function $\mathbf{x} = g(\mathbf{y}, \mathbf{z})$ that transforms the known distribution $p(\mathbf{z})$ to $\mathbf{x}$-space, conditional on $\mathbf{y}$.

The basic building block of the INN is the affine coupling block (ACB). Each ACB consits of four separate fully connected neural networks, denoted as $s_1(\cdot), s_2(\cdot), t_1(\cdot), t_2(\cdot)$. ACBs perform invertible non-linear transformations, i.e. in addition to parametric mappings $f : \mathbb{R}^d \to \mathbb{R}^d$ it also learns the inverse mapping $f^{-1} : \mathbb{R}^d \to \mathbb{R}^d$ for free. Denoting the input vector as $\mathbf{u}$ and the output vector as $\mathbf{v}$, invertibility is achieved by splitting the input vector into two parts $[\mathbf{u}_1, \mathbf{u}_2]$, and performing the following operations on the split input (See Fig.1)

$$\mathbf{v}_1 = \mathbf{u}_1 \odot \exp(s_2(\mathbf{u}_2)) + t_2(\mathbf{u}_2) \tag{15}$$

$$\mathbf{v}_2 = \mathbf{u}_2 \odot \exp(s_1(\mathbf{v}_1)) + t_1(\mathbf{v}_1), \tag{16}$$

where $\odot$ is element-wise multiplication. The outputs $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2)$ are then concatenated and passed to the next ACB. The inverse operation (see Fig.2) is given by

$$\mathbf{u}_2 = (\mathbf{v}_2 - t_1(\mathbf{v}_1)) \odot \exp(-s_1(\mathbf{v}_1)) \tag{17}$$

$$\mathbf{u}_1 = (\mathbf{v}_1 - t_2(\mathbf{u}_2)) \odot \exp(-s_2(\mathbf{u}_2)) \tag{18}$$

This formulation ensures that the Jacobian of the affine transformation is a strictly upper or lower triangular matrix and therefore the determinant is very cheap to compute. The internal functions can be represented by arbitrarily complex neural networks, which need

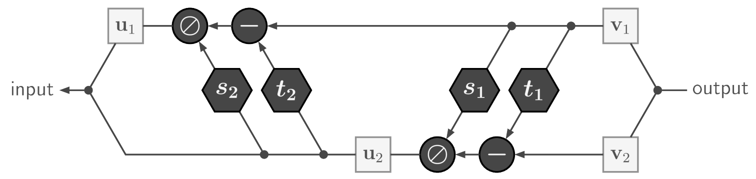

Figure 2: inverse propagation [1]

not to be invertible by themselves.
Since our network is based on [8] it inherits all of its architectural decisions.

3.2 Neural Bayesian Parameter Estimation

Let $\boldsymbol{\theta}$ be the vector of all hidden parameters and $\mathbf{X} := \mathbf{x}_{1:N}$ a multivariate epidemiological time-series. Then, following Bayes theorem, the analytic formula for the posterior is given by

$$p(\boldsymbol{\theta} \mid \mathbf{X}) = \frac{p(\mathbf{X} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\mathbf{X} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}} \tag{19}$$

where $p(\mathbf{X} \mid \boldsymbol{\theta})$ represents the forward model written as the likelihood of observing data $\mathbf{X}$ when the true parameters are $\boldsymbol{\theta}$, $p(\boldsymbol{\theta})$ is the prior distribution encoding our knowledge about plausible parameter combinations, and the denominator is a normalizing constant (the evidence).
During the training phase, the invertible neural network is run in forward mode to learn an accurate model $q(\boldsymbol{\theta} \mid \mathbf{X}) \approx p(\boldsymbol{\theta} \mid \mathbf{X})$ for the posterior distribution of parameters given any observation, using a large number of simulated pairs $(\mathbf{X}_i, \boldsymbol{\theta}_i) \sim p(\mathbf{X} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})$ as training data. In the inference phase, the network operates in inverse direction to estimate the posterior for actually observed data.

3.3 BayesFlow for Epidemiological Inference

The neural architecture is built of three subnetworks: A convolutional filtering network performing noise reduction and feature extraction, a recurrent (LSTM) summary network reducing preprocessed time-series of varying length to statistical summaries of fixed size and an invertible inference network performing Bayesian parameter inference, given the

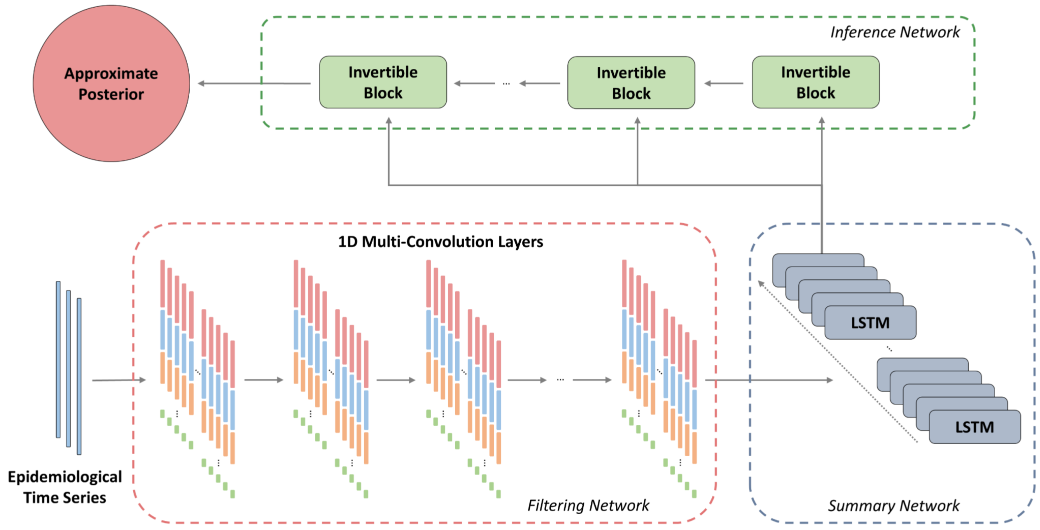

Figure 3: Diagram of the network architecture. First the noisy data is filtrated in a convolutional network. The result is passed to a summary LSTM network. The following statistics are passed to the inference network which then approximates the posterior.

summary of the observations (See Fig 3).

The parameters of all three networks are optimized jointly during the training phase. Denoting the vector of all trainable network parameters $\boldsymbol{\phi}$, the three networks solve the following optimization criterion

$$\hat{\boldsymbol{\phi}} = \arg\min_{\phi} \mathbb{E}_{X\sim p(X)}[\mathbb{KL}(p(\boldsymbol{\theta} \mid \mathbf{X}) \| q_\phi(\boldsymbol{\theta} \mid \mathbf{X})] \tag{20}$$

$$= \arg\min_{\phi} \mathbb{E}_{(X,\theta)\sim p(X,\theta)}[-\log q_\phi(\boldsymbol{\theta} \mid \mathbf{X})] \tag{21}$$

$$= \arg\min_{\phi} \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\|f_\phi(\boldsymbol{\theta}_n; \mathbf{X}_n)\|_2^2}{2} - \log\left|\det \mathbf{J}_{f_\phi}^{(n)}\right|\right) \tag{22}$$

4 Gaussian Process Regression

by Maximilian Richter

In epidemiology the models of interest usually are parametric models $p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta})$, which are used to infer optimal parameter values $\boldsymbol{\theta}$ via maximum likelihood or maximum a posteriori estimation in order to explain data. However, with increasing data complexity, models with a higher number of parameters are needed to explain data reasonably well. For non-parametric models in general the number of parameters depend on the size of the dataset. An important class of non-parametric methods are Gaussian processes, where, instead of inferring a distribution over parameters of a parametric function, these processes can be used to infer a distribution over functions directly. A Gaussian process defines a prior over functions. With the knowledge of measured function values it can then be conditioned into a posterior over functions. The inference of continuous function values is known as Gaussian process regression [4].

4.1 Gaussian Processes

In probability theory a Gaussian process is a special stochastic process $(X_i)_{i\in I}$, for a set of indices I, if every finite subset of its random variables are multivariate normal (Gaussian) distributed. In general, Gaussian processes can be understood as probability distributions over function spaces. A single sample of this distribution is therefore a function with properties restricted by functions of the mean, variance and covariances. For any point $\mathbf{x} \in \mathbb{R}^d$, a Gaussian process assigns a random variable $f(\mathbf{x})$ such that the joint distribution of a finite number of these variables $p(f(\mathbf{x}_i), ..., f(\mathbf{x}_N))$ is itself a Gaussian

$$p(\mathbf{f} \mid \mathbf{X}) = \mathcal{N}(\mathbf{f} \mid \boldsymbol{\mu}, \mathbf{K}), \tag{23}$$

4.2 Regression with Gaussian Processes

Assuming a training dataset with noise-free function values $\mathbf{f}$ at inputs $\mathbf{X}$, a Gaussian process prior can be transformed into a Gaussian process posterior $p(\mathbf{f}_* \mid \mathbf{X}_*, \mathbf{X}, \mathbf{f})$ which can than be used to make predictions $\mathbf{f}_*$ at new inputs $\mathbf{X}_*$. It can be shown that the joint distribution of observed values $\mathbf{f}$ and predictions $\mathbf{f}_*$ is again Gaussian. This distribution can be partitioned into

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} \mathbf{K} & \mathbf{K}_* \\ {\mathbf{K}_*}^T & \mathbf{K}_{**} \end{pmatrix}\right), \tag{24}$$

where $\mathbf{K}_* = k(\mathbf{X}, \mathbf{X}_*)$ and $\mathbf{K}_{**} = k(\mathbf{X}_*, \mathbf{X}_*)$. If the size of the training dataset is N and the set of new input data is N_*, $\mathbf{K}$ is a $N \times N$, $\mathbf{K}_*$ a $N \times N_*$ and $\mathbf{K}_{**}$ is a $N_* \times N_*$ matrix. By the standard rules for conditioning Gaussians, the predictive distribution is given by

$$p(\mathbf{f}_* \mid \mathbf{X}_*, \mathbf{X}, \mathbf{y}) = \mathcal{N}(\mathbf{f}_* \mid \boldsymbol{\mu}, \mathbf{K}) \tag{25}$$

$$\boldsymbol{\mu}_* = \mathbf{K^T}_* \mathbf{K}^{-1} \mathbf{f} \tag{26}$$

$$\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_*. \tag{27}$$

For training data with noise, the predictive distribution can be obtained simply by substituting $\mathbf{y} = \mathbf{f} + \boldsymbol{\epsilon}$, where noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma_y^2 \mathbf{I})$ is independently added to each observation. This implies that $\mathbf{K}_y = \mathbf{K} + \sigma^2 \mathbf{I}$. To include noise ϵ also into predictions $\mathbf{y}_*$, we have to add σ_y^2 to the diagonal of $\boldsymbol{\Sigma}_*$

$$p(\mathbf{y}_* \mid \mathbf{X}_*, \mathbf{X}, \mathbf{y}) = \mathcal{N}(\mathbf{y}_* \mid \boldsymbol{\mu}_*, \boldsymbol{\Sigma}_* + \sigma_y^2 \mathbf{I}) \tag{28}$$

using the definitions for $\boldsymbol{\mu}_*$ and $\boldsymbol{\Sigma}_*$ from Equations (26) and (27), respectively.

4.3 Kernels

A kernel (or covariance function) describes the covariance of the Gaussian process random variables. Together with the mean function $m(\mathbf{x})$, the kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ completely defines the Gaussian process distribution:

$$\mathbf{y} \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}_i, \mathbf{x}_j)). \tag{29}$$

To be a valid kernel function, the resulting covariance matrix $\mathbf{K}$ should be positive definite and hence symmetric. This also guarantees that the kernel matrix is invertible.

There are many possible candidates as kernel functions (see [4]) but we restrict our portrayal to those four used in our analysis of the COVID-19 dataset.

- The most common kernel is the squared exponential or radial basis function (RBF) kernel
$$k_1(\mathbf{x}_i, \mathbf{x}_j) = \sigma_1^2 \exp\left(-\frac{1}{2l_1^2}(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j)\right) \tag{30}$$
with σ_1^2 the overall variance (or amplitude) and l_1 the length scale. This kernel will result in a smooth prior on functions samples from the Gaussian Process and is used to model medium term irregularities (different waves of infection).

- A linear kernel is used to account for the overall trend
$$k_2(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - c)^T(\mathbf{x}_j - c) \tag{31}$$
with c a constant shift.

- The periodic kernel
$$k_3(\mathbf{x}_i, \mathbf{x}_j) = \sigma_3^2 \exp\left(-\frac{2}{l_3^2}\sin^2\left(\pi\frac{(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j)}{p}\right)\right) \tag{32}$$
with σ_3^2 overall variance, l_3 the length scale and p the period. This is used to model the weekly modulations.

- Finally, as already discussed above, the white noise kernel
$$k_4(\mathbf{x}_i, \mathbf{x}_j) = \sigma_4^2\mathbf{I} \tag{33}$$
with σ^2 the variance of the noise, represents the independent and identically distributed noise added to the Gaussian process distribution.

Together, the prior on $\mathbf{y}$ as a function of time is

$$f(t) \sim \mathcal{GP}(\mathbf{0}, k_1(\mathbf{x}_i, \mathbf{x}_j)) + \mathcal{GP}(\mathbf{0}, k_2(\mathbf{x}_i, \mathbf{x}_j)) + \mathcal{GP}(\mathbf{0}, k_3(\mathbf{x}_i, \mathbf{x}_j)) + \mathcal{GP}(\mathbf{0}, k_4(\mathbf{x}_i, \mathbf{x}_j)) \tag{34}$$

Parameter	Prior Distribution
l_1	InverseGamma(4, $N/4$)
l_3	InverseGamma(4, N)
c	Normal(0, 0.05)
σ_1	HalfNormal(0.1)
σ_2	HalfNormal(0.01)
σ_3	HalfNormal(0.2)
σ_4	HalfNormal(0.02)

Table 4: Hyperparameters and their respective prior distributions used in the analysis of the COVID-19 data

4.4 Bayesian Hyperparameter Optimization

Although Gaussian processes are parameter-free methods by definition, the kernel functions have important free hyperparameters, which need to be fine-tuned in order to obtain reasonable results from the posterior. The optimal values for the hyperparameters can be found by grid search, which, unfortunately, scales with the power of the number of parameters. A more suitable method is Bayesian optimization. In the Bayesian framework, the objective to find optimal parameters is the maximization of the marginal likelihood, or equivalently minimization of the negative logarithm of the marginal likelihood, which, for a Gaussian distribution, is given by

$$\log p(\mathbf{y} \mid \mathbf{X}) = \log \mathcal{N}(\mathbf{y} \mid \mathbf{0}, \mathbf{K}_y) = -\frac{1}{2}\mathbf{y}^T \mathbf{K}_y^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K}_y| - \frac{N}{2} \log(2\pi). \tag{35}$$

This is commonly known as maximum a posteriori (MAP) estimation, given non-uniform, informative prior distributions. Table 4 lists the prior distributions for the parameters of our kernel functions.

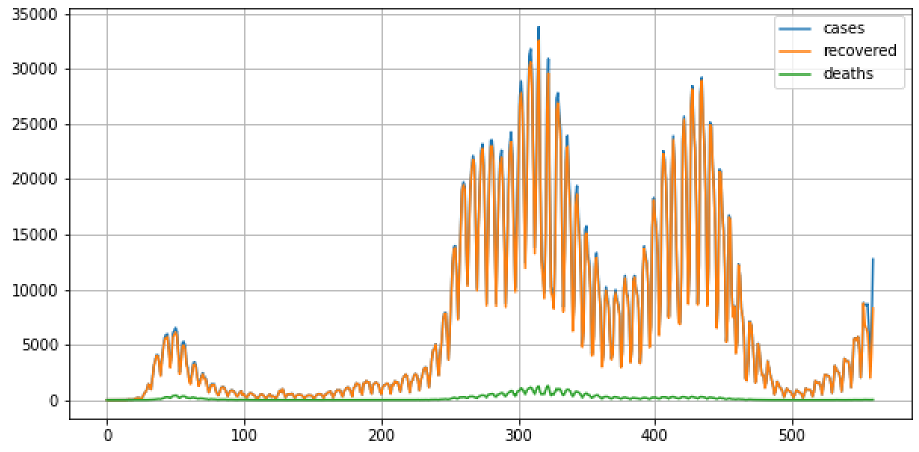

Figure 4: A plot of the COVID-19 reported cases and deaths and its difference as the numbers of recovered cases till 2021-09-01

5 Results

5.1 Data

by Leonard Marks

The data set which is used is given by the Kaggle data on COVID-19 tracking in Germany (`https://www.kaggle.com/headsortails/covid19-tracking-germany`). It is constructed by daily reports on cases of infections and deaths due to corona, as there is the possibility of delay of reports, the data set is corrected continuously, but which does not cancel out the error source of delayed reports mentioned in section 2. Furthermore Kaggle only makes a date entry if either a death or infection case is reported, this could miss out dates on which nothing was reported, so the data set needs to be filled with 0 entries for missing dates. Although it is not mentioned in the column description, the numbers of recovered cases per day is the relative number of recovered people by subtracting the reported deaths on a day from the number of cases on the same (Table 4). As a result, this data set does not give enough insight into the transition from Carriers C and Infected I to Recovered R. So to fit the data set more realistically, we will use the

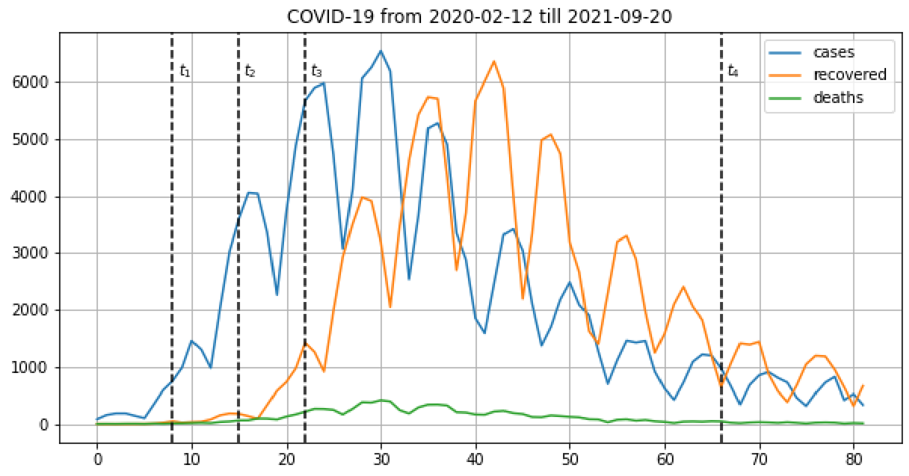

Figure 5: A plot of the COVID-19 reported cases and deaths over a certain time period and the time points t_i

approximated amount of time a person will die of COVID-19 $t_{\text{death}} = 9$ and the official time period after which a person who was infected to stay in quarantine $t_{\text{quarantine}} = 12$. With this we can approximate the number of recovered $R(t)$ people by:

$$R(t) = I(t - t_{\text{quarantine}}) - D(t - t_{\text{death}}) \tag{36}$$

giving us the altered table 5. As it is hard to come by official data with the official numbers of reported recovered cases, the decision falls on this simple approximation.

5.2 BayesFlow

by Leonard Marks

The neural network used is the same as in [8] with 100 epochs. This model yields good predictions for the Kaggle data set, especially for the cumulative and newly infected and (approximated) recovered cases, though the cumulative death cases still lie in the uncertainty boundary, as long as our prediction does not exceed the time point where a second wave is about to unroll. (Figures 9 and 10). The prediction always seems to expect that the increase of cases should drop to 0 for all three.

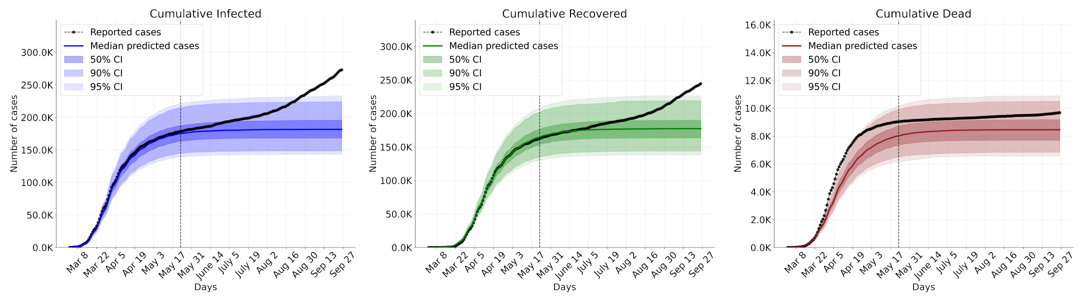

Figure 6: Model prediction of cumulative cases for the first wave

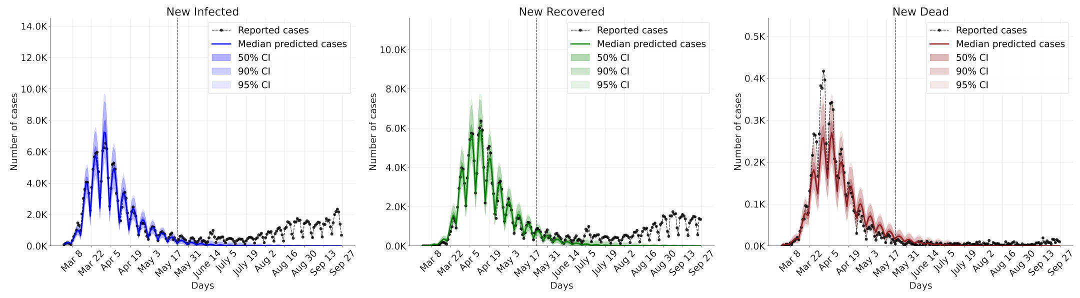

Figure 7: Model prediction of new cases for the first wave

Our parameter estimates are almost identical to the computed ones in [8]. The most interesting differences are μ - Rate at which symptomatic individuals recover, compared to the other results, as well as our prior and posterior, do not fit at all and D_R - Reporting delays for recovered, is the opposite giving an shorter delay then the prior suggests. Furthermore we decided on using the same model for the second COVID-19 wave. For this timer-series the model produces unusable predictions as the simulations do not correspond well with the training data.

5.3 Gaussian Process Regression

by Maximilian Richter

As a baseline for our evaluation we used a pymc3 implementation of Gaussian Processes for epidemiological time-series (`https://github.com/luisroque/bayesian_time_series`). As suggested there, we first checked the prior distribution of functions with hyperparameters for kernel functions discussed in section 4.3. As the scales and ranges seem to fit

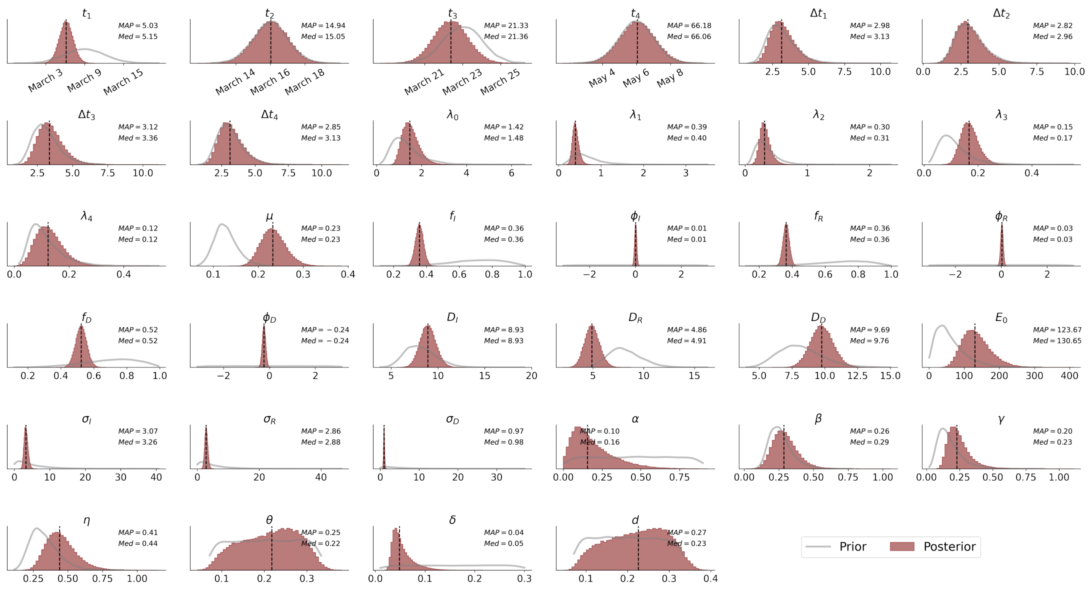

Figure 8: Marginal parameter posteriors

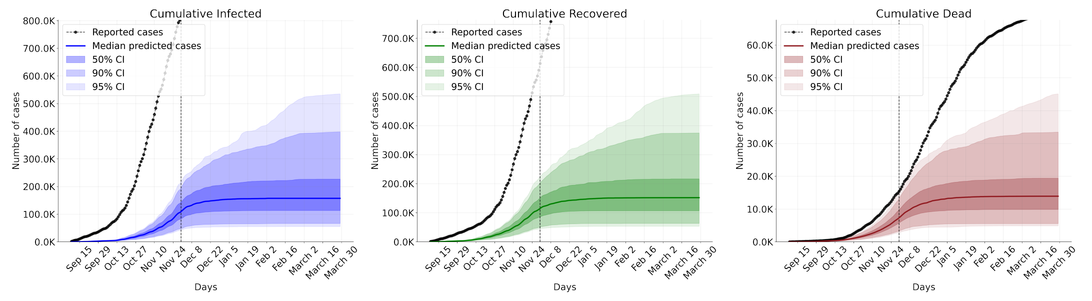

Figure 9: Model prediction of cumulative cases for the second wave

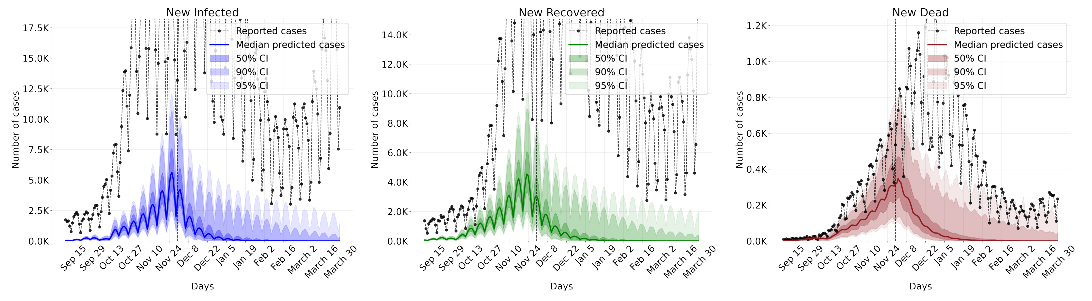

Figure 10: Model prediction of new cases for the second wave

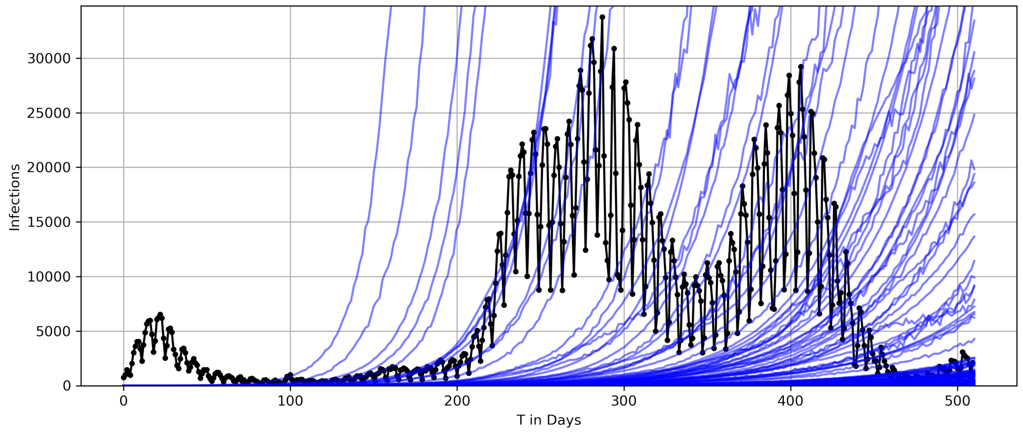

Figure 11: Caption

our dataset by eye, we continued to condition the Gaussian process to our time-series. Results from the prior check can be seen in Figure 11. We have evaluated Gaussian process regression for all three main waves (> 500 days) for 2000 function samples of the posterior distribution. In order to find the optimal hyperparameter values we used 20000 optimization steps via maximum a posteriori estimation, discussed in section 4.4. The whole execution was rather fast, as it took 15 minutes on average for the program to terminate.

As one can see in Figure 12, the Gaussian process is able to catch the full complexity of the data while keeping the uncertainty low. The weekly modulations from the reporting delays gets successfully catched by periodic kernel. We were also able to model the full range of the data with multiple waves of different amplitudes, since the RBF kernel handles mid-frequency modulations very well.

Considering the ability of Gaussian processes to forecast time-series, we conditioned with only 510 days, while keeping the last 48 time points out as test data. As one can see in Figure 13 for approximately 30 days, the forecasting catches the fourth wave considerably well in its 50% confidence intervals. As the prediction gets further into the future, the uncertainty grows also, as expected.

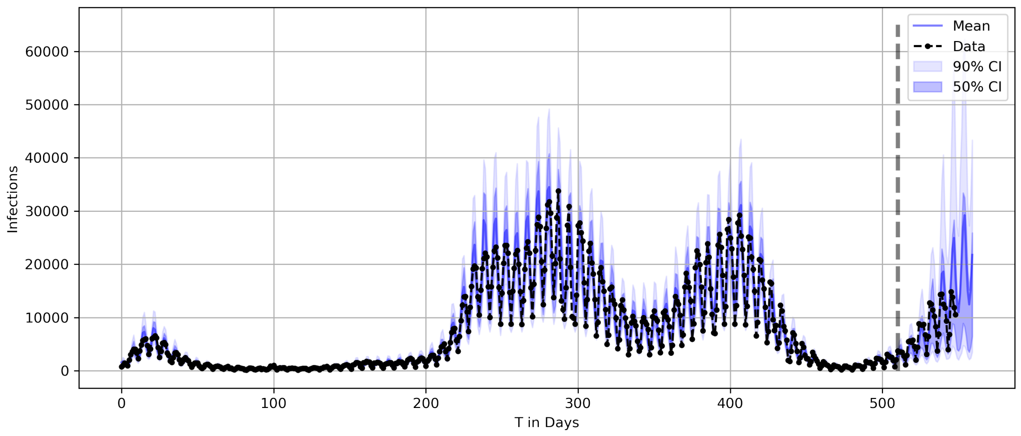

Figure 12: Caption

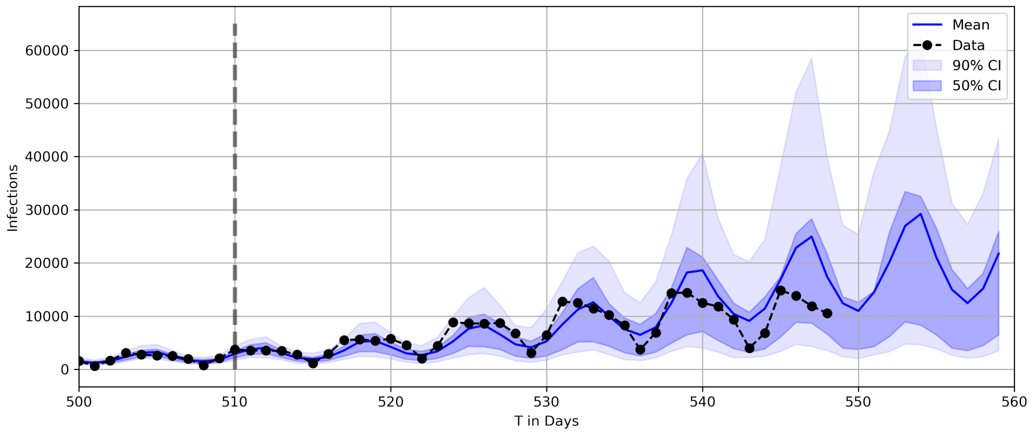

Figure 13: Caption

6 Discussion

by Maximilian Richter (and Leonard Marks)

Against our expectation, it was rather difficult to fit the simulations with parameters sampled from the approximated posterior from the BayesFlow network to the Kaggle dataset. We oberved that the BayesFlow network is indeed a very powerful method with which one can easily learn posterior distributions of model parameters. However, this power is significantly restricted by the choice of the model and perhaps even more importantly by the choice of the priors. Minor changes in the shape of the priors made it nearly impossible for the INN to explain the data reasonably well. This claim was further backed when we tried to train the network on more than the originally 82 days used from the start of the pandemic, as well as the whole range of the data. Since the SEICR-model, used in the paper which we used as a starting point, is not capable of modeling multiple waves of infections, it was merely possible to model the first wave of infections. Being bound by a model is therefore always coupled with strong assumptions about the data generating process and demands great caution when applied naively. There has been research on modeling second-wave dynamics with SEIR-models [5], but these models consist of more than one simulation for different parts of the data (i.e. the respective waves).

Further limitation in our evaluation is the choice of the Kaggle dataset. As described in section 5.1 we are convinced that the raw dataset is not applicable to the epidemiological model we used, since the recovered cases do not shift with the recovery rate of infected people. Processing the data is potentially as source of systematic error and should be avoided in general. Unfortunately, as we tried to at least recreate the first wave of infections with the Kaggel dataset, shifting the recovered and dead people was the only solution for successfully inferring simulation parameters from data. However, this lead to the shift of the posterior μ compared to its prior.

On top on the infection dynamics there is additionally the intervention model, which should lower the infection rates at several time steps. It has shown to be difficult to change the priors of the intervention model especially, since the resulting simulation changes drastically with different values for the infection rate ratio. It was also due to the lack of information on the change of political interventions and their real effectiveness in other times than the first wave in germany that prohibited us from exploring the other infection waves. This contradicts the claim of the authors of [8] that the BayesFlow network is able to operate amortized. Retraining the network for other time-series lengths and other waves would in principle be possible but this would require new considerations on the priors. Training the network on a much wider range for the model parameters and for different time-series lenghts, the BayesFlow network should nonetheless be able to infer parameters from data with varying length and scales. Unfortunately, this would take much more time for the network to converge and was therefore left out in our analysis.

In contrast to that, the non-parametrical Gaussian process regression does not suffer from the same inconveniences encountered while trying to fit the BayesFlow to the data. However, the Gaussian process does not carry nearly as much information, due to the parameter-free formulation. The assumption that the underlying mechanism of the epidemiological time series is a Gaussian process gives no quantification or important interpretations of the model parameters (e.g. rate at which symptomatic individuals recover) which are key to make rational decisions for political intervention and containment of infection cases.

For time-forecasting, the Gaussian processes have shown to be more flexible and easier to handle than the BayesFlow network. This comes at the price that, as mentioned above, the prediction lacks true interpretability. However, we could show that, opposed to the BayesFlow network, the functions sampled from the GP posterior do explain future data very well inside of the 90% confidence intervals. This makes Gaussian process appealing when only future predictions up to 30 days (or times steps in general) are desired. In all other important cases, modeling the data with simulations and infering the parameters with invertible neural networks is still preferable. The huge advantage of invertible neural networks is their interpretability. This is a very crucial ingredient for neural networks used in scientific inference.

Overall we are satisfied with both methods, since both have strengths complementary to the other.

References

[1] Analyzing inverse problems with invertible neural networks. https://hci.iwr.uni-heidelberg.de/vislearn/inverse-problems-invertible-neural-networks/. Accessed: 2021-09-27.

[2] Sir modeling. http://people.wku.edu/lily.popova.zhuhadar/. Accessed: 2021-09-27.

[3] L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E. W. Pellegrini, R. S. Klessen, L. Maier-Hein, C. Rother, and U. Köthe. Analyzing inverse problems with invertible neural networks, 2019.

[4] Q. Y. T. C. J. L. T. D. Fang, X. Zhang. A novel method for carbon dioxide emission forecasting based on improved gaussian processes regression. 2016.

[5] D. Faranda and T. Alberti. Modeling the second wave of covid-19 infections in france and italy via a stochastic seir model. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(11):111101, 2020.

[6] H. Hethcote. The basic epidemiology models: Models, expressions for r0, parameter estimation, and applications. 2008.

[7] S. T. Radev, U. K. Mertens, A. Voss, L. Ardizzone, and U. Köthe. Bayesflow: Learning complex stochastic models with invertible neural networks, 2020.

[8] S. C. N. T. M. V. M. E. T. B. S. T. Radev, F. Graw and U. Köthe. Model-based bayesian inference of disease outbreak dynamics with invertible neural networks an application to the covid-19 pandemics in germany. 2020.