Master's thesis

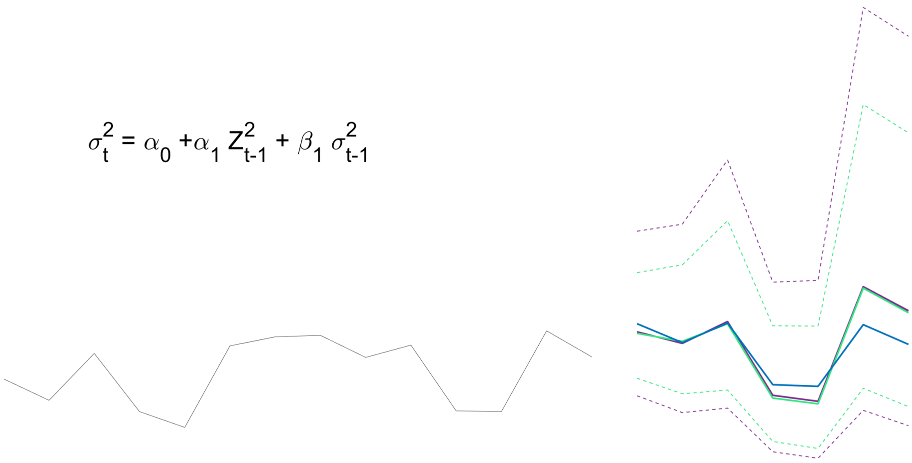
NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Eivind Hagemann Brataas

Statistical Machine Learning on Covid-19 Time Series using Econometrics

Master's thesis in MSMNFMA
Supervisor: Gunnar Taraldsen
Co-supervisor: André Voigt
June 2022

$$\sigma_t^2 = \alpha_0 + \alpha_1 Z_{t-1}^2 + \beta_1 \sigma_{t-1}^2$$



NTNU
Norwegian University of
Science and Technology

Eivind Hagemann Brataas

Statistical Machine Learning on Covid-19 Time Series using Econometrics

Master's thesis in MSMNFMA
Supervisor: Gunnar Taraldsen
Co-supervisor: André Voigt
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

NTNU
Norwegian University of
Science and Technology

Kunnskap for en bedre verden

DEPARTMENT OF MATHEMATICAL SCIENCES

MA3911 - MASTER'S THESIS IN MATHEMATICAL SCIENCES

Statistical Machine Learning on Covid-19 Time Series using Econometrics

Author:
Eivind Hagemann Brataas

Primary supervisor:
Gunnar Taraldsen

Secondary supervisor:
André Voigt

June, 2022

Abstract

This thesis compares three models for forecasting daily new cases for Covid-19. The first model is a class of machine learning models, called an CNN-LSTM model, and was the state-of-the-art model for the stated task on the global daily new cases data set during the summer of 2021. Some months earlier, Taraldsen published two so-called toy models for forecasting the daily new cases in Norway. Both these models are SARIMA models, but one of them assumes Gaussian white noise, while the other assumes that the noise is conditionally heteroscedastic and is modeled by a GARCH model. They both gave accurate predictions and come with a prediction interval, as opposed to the CNN-LSTM model. Additionally, they were much less computer intensive, with only three and four parameters, respectively. On the other hand, the CNN-LSTM model must fit more than 300000 parameters. The models had not yet been applied to other Covid-19 data sets than what was used in their respective articles. This thesis compared the performance of the three models on the global time series data, as well as the Norwegian data. A lot more data have become available since the articles were published. This makes it possible to compare the models on other partitions of both the data sets, and to experiment with reduced sample sizes across all these partitions. Finally, a parametric bootstrap experiment was conducted to get a grasp of the uncertainty in the forecast from the CNN-LSTM model. While the CNN-LSTM model achieved some accurate forecasts, the results of this thesis suggest that the SARIMA model with GARCH noise may be the model of choice for the earliest parts of the data sets, while the SARIMA model with Gaussian white noise would be the best choice on the rest of the data sets, where its predictions are more accurate and has about the same spread. These results are mostly explained by the varying heteroscedasticity of the two time series.

Preface

I would like to preface this by thanking the people that made the creation of this thesis possible.

I would first like to express my deepest gratitude to my supervisor Gunnar Taraldsen. Gunnar introduced me to the interesting field of time series two years ago. We have since then shared many good discussions, and he has kept me motivated. With his great theoretical knowledge, he has always kept the big picture in mind. I would like to thank him for devoting a lot of his time and for being patient with me. Without him, the thesis would not have existed.

I would also like to thank my secondary supervisor Andre Voigt for sharing his insights about the application of the models in the real world.

Finally, I want to thank all my friends, my family, and my partner Elin for supporting me while writing this thesis.

The thesis was exiting to work on, as it includes topics from applied and theoretical statistics, to analysis, to machine learning. It was therefore the perfect ending to my time as a master student.

Table of Contents

List of Tables

List of Figures

1 Introduction

The novel corona virus 2019 (Covid-19) took the world by storm, killing over 6 million people in two years. Unlike previous pandemics, lots of data have been gathered about the decease. This enables accurate forecasts of the spread of the virus, which have been able to function as an early warning system for authorities to react with appropriate responses (Dehning et al., 2020). Even though the current pandemic might be under control, globalisation will likely cause future viruses with the same potency. Thus, the development of better forecasting tools for decease spread should continue, to hopefully prevent another disaster. There are at least two schools of thought when it comes to modeling the spread of deceases. The epidemiological approach attempts to understand the inner workings of the virus, which can then be used to model its spread, while a data-driven approach ignores where the data comes from and purely uses the data at hand. In its simplest form, the latter approach only considers the new Covid-19 cases X_t at day t. After collecting the observations $x_1, \ldots, x_n$, the goal is to predict the amount of future new cases $X_{n+1}, \ldots, X_{n+h}$ as accurately as possible. This thesis focuses on solving this exact problem.

In an article published by Zain et al. (2021), several predictive models were trained to solve the above problem. It included a proposed machine learning model that combined the abstraction capabilities of a Convolutional Neural Network (CNN) with a Long-Short Term Memory (LSTM) model's ability to learn long term dependencies. The current state-of-the-art model at the time of publishing was also among the compared models. From their empirical results on the global daily new cases data set from January 4th to August 14th 2020, they concluded that the proposed model was the best model for the task at hand. This conclusion was based on the accuracy of 28-day forecasts from each model. Also among the most accurate models in this study was an Auto Regressive Moving Average (ARMA) model. The ARMA model is a large class of models, with many possible alterations. None of these alterations were explored in their article. Some months earlier, Taraldsen (2020b) published two so-called toy models for the Norwegian daily new cases from February 21st to November 10th 2020. They were both Seasonal Auto Regressive Integrated Moving Average (SARIMA) models, but one of them assume that the underlying noise is Gaussian and independent of time, while the other one assume GARCH$(1, 1)$ noise; a more general noise model that allows conditional heteroscedasticity. On the data set used by Taraldsen, both these models delivered accurate forecasts. For the daily new cases of Covid-19 in some of the key countries, SARIMA models and ARMA-GARCH models both have been shown to give accurate forecasts (Kumar et al., 2021)(Ekinci, 2021). However, at the time of writing, a combined SARIMA-GARCH model for daily new cases have yet to be tested on the global new cases data. In this thesis, the three models were replicated as precisely as possible, and their performances were compared on both the global and the Norwegian data set, seen in Figure 1. The articles were written based on data from 2020. Since then, a lot more data has become available. The two time series therefore include the daily new cases up to February 20th 2022. A lot has happened to the handling of the virus over the course of the two years: new variants have emerged, vaccines have been developed, and rulings and attitudes surrounding the decease have evolved. How will these changes affect the models fit and prediction capabilities? And what happens to the results if the sample size is reduced? This thesis attempts to answer these questions.

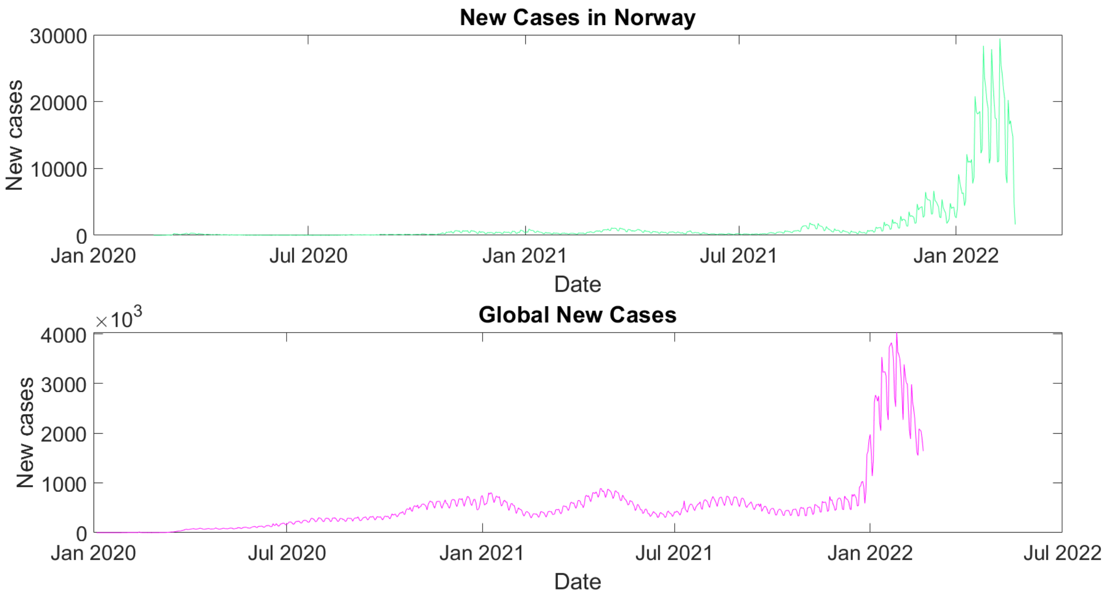


Figure 1: The Norwegian and global Covid-19 new cases time series.

The replicated models were initially tested on the two data sets from the articles from Zain et. al. and Taraldsen. This made the results from the models in the thesis comparable to the results from the articles. The models were later tested on newer and older partitions of the data sets, by regulating the last date included in the training set, and the number of samples to include preceding said date. The primary goal was to compare the accuracy of the models across these partitions of the data sets. However, this was not the only task. If the models were to be used in practice, the uncertainty of the forecasts is almost as important as the forecast itself. When using Taraldsen's SARIMA models, this uncertainty can easily be approximated with theoretical 95% prediction intervals. The CNN-LSTM model has no way to generate such an interval. To get a grasp of the spread of this model's forecast, a small parametric bootstrap study was conducted. The secondary goal of the thesis was to describe and compare the spread in the forecasts of each model. The two SARIMA models only contain 3 and 4 parameters, respectively. Some of these specifies the structure of the auto regressive components and the moving average components, while others describe the noise component of the models. The third goal of the thesis was to investigate the behaviour of these parameters, especially the ones connected to the noise of the models. The CNN-LSTM model contains more than 300000 parameters. Such an investigation was therefore not feasible for this model.

The time series analysis made in this thesis require some baseline knowledge about ARMA models and machine learning models. The necessary theory is given in Section 2. Details about the specific models and how the forecasts in this thesis were constructed are given in Section 3. The models were then applied to the two data sets, as described above. These results are presented in Section 4 and discussed in Section 5. The final thoughts and conclusions in Section 6 finalize the thesis.

2 Theory

This section introduces the necessary concepts and theoretical results needed to understand the thesis and demonstrates how the later results were generated.

2.1 Time Series

The **random variables** lay the foundation of statistical theory. The exact definition is given.

Definition 2.1 (Random variable). *Let* $(\Omega, \mathcal{F})$ *and* $(\tilde{\Omega}, \tilde{\mathcal{F}})$ *be measurable spaces.*

A map $X : \Omega \to \tilde{\Omega}$ *is called a random variable if*

$$X^{-1}(\tilde{F}) \in \mathcal{F} \quad \text{for all } \tilde{F} \in \tilde{\mathcal{F}}.$$

In probability theory, the measurable spaces $(\Omega, \mathcal{F})$ and $(\tilde{\Omega}, \tilde{\mathcal{F}})$ are called event spaces. In this thesis, $\tilde{\Omega} = \mathbb{R}$ and $\tilde{\mathcal{F}} = \mathcal{B}(\mathbb{R})$, the Borel σ-algebra. The associated probability space (measure space) is $(\Omega, \mathcal{F}, P)$, where P is the cumulative distribution function of X.

The Covid-19 new-cases data are viewed as a series of events from random variables X_t, one for each passing day t. A realization of these random variables is $x_1, \ldots, x_n$. Figure 1, shows one such realization for each of the two time series.

Definition 2.2 (Time series). *A time series is a sequence of random variables* $\{X_t\} = \{X_t\}_{t \in \mathbb{R}}$ *indexed by time. In this thesis, a realization of X, denoted by* $(x_1, \ldots, x_n)$*, is also called a time series.*

More general definitions of time series exist. However, for the purposes of this thesis, the above definition will suffice.

The remainder of this section outlines how to model time series. For the statistical models, a desired property for time series when modeling is *week stationary*. Essentially, the elements of time series with this property maintains the same relationship to its neighbouring data points across the whole time series.

Definition 2.3 (Weak Stationarity). *A time series X is weakly stationary if it is shift-invariant. That is,* X_t *has a constant mean for all t and* $cov(X_i, X_j) = h(|i, j|)$ *for some unsigned function h.*

In other words, the correlation between two variables of a weakly stationary time series is only dependent on the distance between their respective points in time, not time itself. On the other hand, some time series has no correlation between the neighbouring variables. These are called white noise processes.

Definition 2.4 (White noise). *If* $Z \sim WN(0, \sigma^2)$*, then* $\{Z_t\}$ *is a sequence of uncorrelated random variables that all have mean zero and the same variance* σ^2*. This sequence is called a white noise process.*

2.2 Defining the ARMA process and its extensions

The **Auto Regressive Moving Average** (ARMA) process is widely used to model weakly stationary time series. It is a general class of processes that can be specified by the hyper parameters p and q.

Definition 2.5 (ARMA(p, q) process). *Let* $\{X_t\}$ *be a zero-mean weekly stationary time series.* $\{X_t\}$ *is an ARMA*(p, q) *process if it is a weakly stationary solution to the equations*

$$\phi(B)X_t = \theta(B)Z_t, \tag{1}$$

where $\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots \phi_p B^p$, $\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q$ *and B is the back-shift operator defined by* $BX_t = X_{t-1}$*. Additionally,* $Z \sim WN(0, \sigma^2)$ *and* $\phi(B)$ *and* $\theta(B)$ *have no common factors.*

The special cases ARMA$(p, 0)$ *and ARMA*$(0, q)$ *are the AR*(p) *and MA*(q) *process, respectively.*

For most time series, $E[X_t] = \mu \neq 0$. However, the process $\{X_t - \mu\} \sim \text{ARMA}(p,q)$. These processes are called an ARMA process with mean μ. The remaining theoretical results only considers zero-mean time series without any loss of generality.

The stationarity assumption of the solution of the ARMA model implies some restriction of the possible solutions of Equation (1). If $\phi(z) \neq 0$, a candidate solution is

$$X = \frac{\theta(B)}{\phi(B)} Z_t \tag{2}$$

Let $\psi(z) = \frac{\theta(z)}{\phi(z)} = \psi_0 + \psi_1 z + \psi_2 z^2 + \cdots$. The solution $X_t = \psi_0 + \psi_1 Z_{t-1} + \psi_2 Z_{t-2} + \cdots$ is zero-mean since it is a linear combination of zero-mean random variables. To satisfy the definition of week stationarity, the variance of the solution X_t needs to be finite. However, for $h \geq 1$,

$$\begin{aligned} E[X_t X_{t-h}] &= \psi_0^2 + \psi_0\psi_1(E[Z_t] + E[Z_{t+h}]) + \cdots + \psi_h\psi_0 E[Z_{t-h}Z_{t-h}] + \cdots \\ &= \psi_0^2 + \sum_{i=h}^{\infty} \psi_i \psi_{i-h} E[Z_{t-i}^2] = \psi_0^2 + \sigma^2 \sum_{i=h}^{\infty} \psi_i \psi_{i-h}, \end{aligned}$$

which may diverge if $\psi(z)$ is not absolutely summable. Assuming absolute summability of $\psi(z)$ is sufficient to obtain a stationary solution of the ARMA equations. This would also imply uniqueness of the solution. However, to verify that an ARMA process satisfies this condition can take a lot of time, which makes this criterion impractical. An equivalent, and more practical condition is the following:

$$\phi(z) \neq 0, \quad \textit{for all } z \in \mathbb{N}$$

i.e., $\phi(z)$ has no zeros on the unit circle. To understand why this is equivalent, consider the Taylor expansion of $\frac{1}{\phi(z)}$ around zero. For $\delta > 0$,

$$\frac{1}{\phi(z)} = \sum_{i=-\infty}^{\infty} a_i z^i < \infty, \quad \text{for } 1 - \delta < z < 1 + \delta.$$

Since z can be made arbitrarily close to 1, the sum $\sum_{i=-\infty}^{\infty} |a_i|$ converges. Applying $\frac{1}{\phi(B)}$ to both sides of the ARMA equations shows that

$$X_t = \frac{1}{\phi(B)} \phi(B) X_t = \frac{1}{\phi(B)} \theta(B) Z_t = \psi(B) Z = \sum_{i=-\infty}^{\infty} \psi_i Z_{t-i}.$$

Thus, $\psi(B)Z$ solves Equation (1) if and only if $\phi(z) \neq 0$ *for all* $z \in \mathbb{N}$. The above argument shows that in most cases, the ARMA process can be expressed as a function of all the past, present and future noise components. In certain cases, X_t can be expressed uniquely using only Z_{t-h} for $h \leq 1$.

Definition 2.6 (Causal process)**.** *The process $\{X_t\}$ is causal if the polynomials $\theta(z)$ and $\phi(z)$ have no common zeros and there exists a sequence $\{\psi_j\}_{j=0}^{\infty}$ with $\sum_{j=0}^{\infty} |\psi_j| < \infty$ such that*

$$X_t = \sum_{j=0}^{\infty} \psi_j Z_{t-j}, \quad \textit{for all } t.$$

In other words, a causal process is independent of future noise. For an ARMA(p,q) process, this is equivalent to $\phi(z)$ having no zeros inside the unit circle (Brockwell et al., 2016, p. 75). Note that $\sum_{j=0}^{\infty} \psi_j Z_{t-j}$ is an MA(∞) process. Thus, any causal ARMA(p,q) process can be expressed as a MA(∞) process.

The dual property for $\{Z_t\}$ is called invertibility.

Definition 2.7 (Invertible process)**.** *The process* $\{X_t\}$ *is invertible if the polynomials* $\theta(z)$ *and* $\phi(z)$ *have no common zeros and there exists a sequence* $\{\pi_j\}_{j=0}^{\infty}$ *with* $\sum_{j=0}^{\infty} |\pi_j| < \infty$ *such that*

$$Z_t = \sum_{j=0}^{\infty} \pi_j X_{t-j}, \quad \textit{for all } t.$$

Equivalently, an invertible process is a process where the noise only depends on past observations. The invertibility property of an ARMA process can be shown to be equivalent to all zeros of $\theta(z)$ being outside the unit circle (Brockwell et al., 2016, p. 76).

Both these properties are highly desirable, as they simplify some calculations. One of them is the derivations surrounding large-sample prediction with ARMA models. Luckily, it can be shown that a non-causal or non-invertible ARMA process can be transformed into a causal and invertible ARMA process by finding a new white noise sequence $\{W_t\}$. Hence, without loss of generality, all future ARMA processes are assumed to be causal and invertible (Brockwell et al., 2016, p. 50). Note that for an MA(q) process, $\psi(z) = \theta(z)$, whereas for an AR(p) process, $\pi(z) = \phi(z)$. In other cases, the coefficients ψ_i and π_i can be found by comparing the sides of the following equations:

$$\psi(z)\phi(z) = \theta(z)$$
$$\pi(z)\theta(z) = \phi(z).$$

The assumption that $\{X_t\}$ is weakly stationary, is important to note. In the real world, this is not always the case. For example, both time series seen in Figure 1 have upward trends, implying that the mean of these series are not constant. Even if $\{X_t\}$ is non-stationary, it is often possible to transform it to a stationary time series. The remainder of this section describe how small alterations to the ARMA process can be made to address time series with trend and seasonality.

Definition 2.8 (The ARIMA(p, d, q) process)**.** *The time series* $\{X_t\}$ *is a ARIMA*(p, d, q) *process if*

$$(1 - B)^d X_t$$

is a causal ARMAp, q) *process.*

The ARIMA(p, d, q) model can be used to transform a series with a linear trend to a stationary process, which can then be modeled by an ARMA(p, q) process. The time series for the new Covid-19 cases seems to both have a trend and a seasonal component (of period 7). This time series would not be stationary by applying a single difference operation. The **Seasonal Auto Regressive Integrated Moving Average** (SARIMA) model aims to remove both the trend and the seasonal component of $\{X_t\}$ by differencing not only in the first order, but also for order s, where s is the period of the time series.

Definition 2.9 (The SARIMA process)**.** *The time series* $\{X_t\}$ *is a SARIMA*$(p, d, q) \times (P, D, Q)_s$ *process if the differenced series* $Y_t = (1 - B)^d (1 - B^s)^D X_t$ *is an ARMA process of the of the following form*

$$\phi(B)\Phi(B^s)X_t = \theta(B)\Theta(B^s)Z_t, \tag{3}$$

where $\phi(z) = 1 - \phi_1 z - \cdots - \phi_p z^p$, $\Phi(z) = 1 - \Theta_1 z - \cdots - \Theta_P z^P$, $\theta(z) = 1 + \theta_1 z + \cdots + \theta_q z^q$ *and* $\Theta(z) = 1 + \Theta_1 z + \cdots + \Theta_Q z^Q$.

Note that Equation (3) can be rewritten as an ARMA(pP, qQ) process (possibly with some of the coefficients being fixed to zero and others as products of previous coefficients). Future results will only consider ARMA processes, as these are notionally simpler, but these also apply to SARIMA models.

2.3 The GARCH process

For all processes defined up to this point, the mean of X_t has been conditionally dependent on its previous iterations, while the variance σ^2 has stayed constant. Sometimes it is reasonable to assume that not only the mean, but also the variance of $\{X_t\}$ changes over time. The noise component can be modeled by its own process. A popular choice is the **Generalized Auto Regressive Conditional Heteroscedasticity** (GARCH) process, developed by Bollerslev (1986). It allows the variance at the current time step to depend on previous variances and the previous iterations of Z_t.

Definition 2.10 (GARCH(p, q) process)**.** *The GARCH(p, q) process is the weekly stationary solution to the equations*

$$Z_t = \sigma_t e_t, \tag{4}$$

for $e_t \sim IID(0,1)$*, where* σ_t *is defined by the recursions*

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i Z_{i-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2, \ \forall t, \tag{5}$$

for $\alpha_0 > 0$ *and* $\alpha_i, \beta_j \geq 0$ *for all* i, j*.*

Under these assumptions, $\sigma_t^2 > 0$ for all t. When modelling, one must assume a particular distribution of e_t. This is often the standard normal distribution or Student's t-distribution. The quantity σ_t^2 is commonly referred to as the volatility of the time series. A time series where the fluctuations are followed by fluctuations of similar magnitude is said to exhibit **persistence of volatility**. In these cases, it might be wise to add a GARCH model to the underlying noise.

2.3.1 The existence of the GARCH(1,1) model

The GARCH$(1,1)$ model, specified in Equation (6), has been particularly popular in time series forecasting.

$$Z_t = \sigma_t^2 e_t, \quad \sigma_t^2 = \alpha_0 + \alpha_1 Z_{t-1}^2 + \beta_1 \sigma_{t-1}^2 \tag{6}$$

However, a weekly stationary solution to Equation (6) only exists for certain values of α_0, α_1 and β_1. Observe that

$$\begin{aligned}
Z_t^2 &= \sigma_t^2 e_t^2 = (\alpha_0 + \alpha_1 Z_{t-1}^2 + \beta_1 \sigma_{t-1}^2) e_t^2 = \alpha_0 e_t^2 + e_t^2 (\alpha_1 e_{t-1}^2 + \beta_1) \sigma_{t-1}^2 \\
&= \alpha_0 e_t^2 + e_t^2 (\alpha_1 e_{t-1}^2 + \beta_1)(\alpha_0 + \alpha_1 Z_{t-2}^2 + \beta_1 \sigma_{t-2}^2) \\
&= \alpha_0 e_t^2 + \alpha_0 e_t^2 (\alpha_1 e_{t-1}^2 + \beta_1) + e_t^2 (\alpha_1 e_{t-1}^2 + \beta_1)(\alpha_1 e_{t-2}^2 + \beta_1) \sigma_{t-2}^2 \\
&= \cdots = \alpha_0 e_t^2 \left(1 + \sum_{k=1}^{n} \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1) \right) + e_t^2 \sigma_{t-(n+1)}^2 \prod_{i=1}^{n+1} (\alpha_1 e_{t-i}^2 + \beta_1).
\end{aligned}$$

This gives a candidate solution to Equation (6), namely

$$\lim_{n \to \infty} Z_{t,n} = e_t \sqrt{\alpha_0 \left(1 + \sum_{k=1}^{\infty} \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1) \right)},$$

where

$$Z_{t,n}^2 = \alpha_0 e_t^2 \left(1 + \sum_{k=1}^{n} \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)\right).$$

However, it must be shown that this limit exists. As will be clear, this is not always the case.

Theorem 2.11. *$Z_{t,n}$ is a Cauchy sequence and converges to*

$$\tilde{Z}_t = e_t \sqrt{\alpha_0 \left(1 + \sum_{k=1}^{\infty} \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)\right)}$$

in L^2 if and only if

$$\alpha_1 + \beta_1 < 1.$$

Proof. To show that $Z_{t,n}$ is Cauchy, one needs to show that $\forall \epsilon \exists N \in \mathbb{N}$ s.t.

$$E[(Z_{t,n} - Z_{t,m})^2] < \epsilon$$

for $n, m \geq N$. Expanding this expression gives that

$$E[(Z_{t,n} - Z_{t,m})^2] = E[Z_{t,n}^2 - 2Z_{t,n}Z_{t,m} + Z_{t,m}^2] = E[Z_{t,n}^2] + E[Z_{t,m}^2] - 2E[Z_{t,n}Z_{t,m}]. \tag{7}$$

Note that

$$\begin{aligned} Z_{t,n}Z_{t,m} &= e_t \sqrt{\alpha_0 \left(1 + \sum_{k=1}^{n} \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)\right)} e_t \sqrt{\alpha_0 \left(1 + \sum_{k=1}^{m} \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)\right)} \\ &= e_t^2 \alpha_0 \sqrt{1 + \sum_{k=1}^{n} \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)} \sqrt{1 + \sum_{k=1}^{m} \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)} \end{aligned}$$

Without loss of generality, assume $n \geq m$. Since all terms in the above equation are positive, $Z_{t,n}Z_{t,m} \geq Z_{t,m}^2$. By monotonicity of the expected value operator, it is clear that

$$E[Z_{t,n}Z_{t,m}] \geq E[Z_{t,m}^2].$$

Plugging this result into Equation (7) gives

$$\begin{aligned} E[(Z_{t,n} - Z_{t,m})^2] &\leq E[Z_{t,n}^2] - E[Z_{t,m}^2] \\ &= E\left[\alpha_0 e_t^2 \left(1 + \sum_{k=1}^{n} \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)\right)\right] - E\left[\alpha_0 e_t^2 \left(1 + \sum_{k=1}^{m} \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)\right)\right] \\ &= \alpha_0 \sum_{k=1}^{n} E\left[e_t^2 \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)\right] - \alpha_0 \sum_{k=1}^{m} E\left[e_t^2 \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)\right] \\ &= \alpha_0 \sum_{k=m}^{n} E\left[e_t^2 \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)\right] \end{aligned}$$

The following lemma can be utilized to reduce the above expression even further.

Lemma 2.12. $E\left[e_t^2 \prod_{i=1}^k (\alpha_1 e_{t-i}^2 + \beta_1)\right] = (\alpha_1 + \beta_1)^k$

Proof. Proof by induction:

$k = 1$:

$$E[e_t^2(\alpha_1 e_{t-1}^2 + \beta_1)] = \alpha_1 E[e_t^2 e_{t-1}^2] + \beta_1 E[e_t^2] = \alpha_1 E[e_t^2]E[e_{t-1}^2] + \beta_1 E[e_t^2] = \alpha_1 + \beta_1$$

Assume it holds for $k = m$. Then

$$\begin{aligned}
E\left[e_t^2 \prod_{i=1}^{m+1}(\alpha_1 e_{t-i}^2 + \beta_1)\right] &= E\left[(\alpha_1 e_{t-m-1}^2 + \beta_1) e_t^2 \prod_{i=1}^{m}(\alpha_1 e_{t-i}^2 + \beta_1)\right] \\
&= E\left[(\alpha_1 e_{t-m-1}^2 + \beta_1) E\left[e_t^2 \prod_{i=1}^{m}(\alpha_1 e_{t-i}^2 + \beta_1) \mid e_{t-m-1}\right]\right] \\
&= E\left[(\alpha_1 e_{t-m-1}^2 + \beta_1) E\left[e_t^2 \prod_{i=1}^{m}(\alpha_1 e_{t-i}^2 + \beta_1)\right]\right] \\
&= E[(\alpha_1 e_{t-m-1}^2 + \beta_1)(\alpha_1 + \beta_1)^m] = (\alpha_1 + \beta_1)^m E[\alpha_1 e_{t-m-1}^2 + \beta_1] \\
&= (\alpha_1 + \beta_1)^{m+1},
\end{aligned}$$

where the second line equations used the law of iterated expectation and the fact that $e_t \sim \mathrm{IID}(0, 1)$.

■

The above lemma implies that

$$E[(Z_{t,n} - Z_{t,m})^2] \leq \alpha_0 \sum_{k=m}^{n} (\alpha_1 + \beta_1)^k.$$

Since α_1 and β_1 are assumed positive, the series converges if and only if $\alpha_1 + \beta_1 < 1$. In this case, the sum can be made arbitrarily small by increasing n and m. Thus, $Z_{t,n}$ is Cauchy and the limit

$$\tilde{Z}_t = e_t \sqrt{\alpha_0 \left(1 + \sum_{k=1}^{\infty} \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)\right)}$$

exists. ■

The following argument shows that $\tilde{Z}_t$ satisfies Equation (6). Let $\tilde{Z}_t = \tilde{\sigma}_t^2 e_t$. Then

$$\tilde{\sigma}_t^2 = \alpha_0 \left(1 + \sum_{k=1}^{\infty} \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)\right) = \alpha_0 \left(1 + \alpha_1 e_{t-1}^2 + \beta_1 + \sum_{k=2}^{\infty} \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)\right).$$

Note that

$$\sum_{k=2}^{\infty} \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1) = (\alpha_1 e_{t-1}^2 + \beta_1) \sum_{k=1}^{\infty} \prod_{i=1}^{k} (\alpha_1 e_{t-1-i}^2 + \beta_1)$$

and that

$$\tilde{\sigma}_{t-1}^2 = \alpha_0 \left(1 + \sum_{k=1}^{\infty} \prod_{i=1}^{k} (\alpha_1 e_{t-1-i}^2 + \beta_1)\right).$$

Shuffling the terms around in the above equation gives

$$\sum_{k=1}^{\infty} \prod_{i=1}^{k} (\alpha_1 e_{t-1-i}^2 + \beta_1) = \frac{\tilde{\sigma}_{t-1}^2}{\alpha_0} - 1.$$

Combining the above equations results in the following:

$$\tilde{\sigma}_t^2 = \alpha_0 \left(1 + \alpha_1 e_{t-1}^2 + \beta_1 + (\alpha_1 e_{t-1}^2 + \beta_1)(\frac{\tilde{\sigma}_{t-1}^2}{\alpha_0} - 1)\right) = \alpha_0 + \alpha_1 e_{t-1}^2 \tilde{\sigma}_{t-1}^2 + \beta_1 \tilde{\sigma}_{t-1}^2 = \alpha_0 + \alpha_1 \tilde{Z}_{t-1}^2 + \beta_1 \tilde{\sigma}_{t-1}^2.$$

Hence $\tilde{\sigma}_t^2$ can be written as in Equation (6), and $\tilde{Z}_t$ is a solution to the GARCH(1, 1) equations. For $\tilde{Z}_t$ to be a valid solution, it needs to be weekly stationary. When $\alpha_1 + \beta_1 < 1$, the unconditional variance

$$\begin{aligned} E[Z_t^2] &= E\left[e_t^2 \alpha_0 \left(1 + \sum_{k=1}^{\infty} \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)\right)\right] = \alpha_0 + \alpha_0 \sum_{k=1}^{\infty} E\left[e_t^2 \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)\right] \\ &= \alpha_0 (1 + \sum_{k=1}^{\infty} (\alpha_1 + \beta_1)^k) = \alpha_0 \sum_{k=0}^{\infty} (\alpha_1 + \beta_1)^k = \frac{\alpha_0}{1 - \alpha_1 - \beta_1} < \infty. \end{aligned}$$

Further, by using the law of total expectation

$$E[Z_t] = E[Ae_t] = E[AE[e_t \mid e_s, s < t]] = E[AE[e_t]] = 0$$

where $A = \sqrt{\alpha_0 \left(1 + \sum_{k=1}^{\infty} \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)\right)}$. By the same argument,

$$E[Z_{t+h} Z_t] = E[B e_{t+h} e_t] = E[BE[e_{t+h} e_t \mid e_s, s < t + h]] = E[BE[e_{t+h} e_t]] = 0,$$

for $h < 0$ and with $B = \sqrt{\alpha_0 \left(1 + \sum_{k=1}^{\infty} \prod_{i=1}^{k} (\alpha_1 e_{t+h-i}^2 + \beta_1)\right)} \sqrt{\alpha_0 \left(1 + \sum_{k=1}^{\infty} \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)\right)}$. These three qualities imply that $Z_t \sim WN(0, \frac{\alpha_0}{1-\alpha_1-\beta_1})$, which is a weekly stationary process.

To demonstrate that the solution is unique, denote $\tilde{\tilde{Z}}_t = \tilde{\tilde{\sigma}}_t e_t$ as another solution. Once again, iterating thought Equation (6) gives

$$\tilde{\tilde{Z}}_t^2 = \cdots = \alpha_0 e_t^2 \left(1 + \sum_{k=1}^{n} \prod_{i=1}^{k} (\alpha_1 e_{t-i}^2 + \beta_1)\right) + e_t^2 \tilde{\tilde{\sigma}}_{t-(n+1)}^2 \prod_{i=1}^{n+1} (\alpha_1 e_{t-i}^2 + \beta_1).$$

When $\alpha_1 + \beta_1 < 1$, the mean square limit

$$\begin{aligned} E\left[(\tilde{\tilde{Z}}_t - \tilde{Z}_t)^2\right] &= E\left[e_t^2 \tilde{\tilde{\sigma}}_{t-(n+1)}^2 \prod_{i=1}^{n+1} (\alpha_1 e_{t-i}^2 + \beta_1)\right] \\ &= E\left[\tilde{\tilde{\sigma}}_{t-(n+1)}^2 E\left[e_t^2 \prod_{i=1}^{n+1} (\alpha_1 e_{t-i}^2 + \beta_1) \mid \tilde{\tilde{\sigma}}_{t-(n+1)}^2\right]\right] \\ &= (\alpha_1 + \beta_1)^{n+1} E[\tilde{\tilde{\sigma}}_{t-(n+1)}^2] \to 0 \end{aligned}$$

as $n \to \infty$. The above derivation used the fact that $E[\tilde{\tilde{\sigma}}^2_{t-(n+1)}] < \infty$ since any solution to Equation (6) has to be weekly stationary. Hence, $\tilde{Z}_t$ is the unique weekly stationary solution to the GARCH$(1,1)$ equations. Here is a summary of the results:

Solution of GARCH(1,1) Equations:
If $(\alpha_1 + \beta_1) < 1$, a weekly stationary solution of the GARCH(1,1) equations is

$$Z_t = e_t \sqrt{\alpha_0 \left(1 + \sum_{k=1}^{\infty} \prod_{i=1}^{k} (\alpha_1 e^2_{t-i} + \beta_1)\right)}.$$

It has the properties

$$E[Z_t] = 0,$$

$$Var(Z_t) = E[Z_t^2] = \frac{\alpha_0}{1 - \alpha_1 - \beta_1}$$

and

$$E[Z_{t+h} Z_t] = 0, \text{ for } h > 0.$$

It can be shown that, in general, a weekly stationary solution to the Equation (4) and Equation (5) exists if and only if $\sum_{i=1}^{p} \alpha_i + \sum_{j=1}^{q} \beta_j < 1$ (Bollerslev, 1986).

Remark: The above derivations only showed that Z_t is unconditionally independent of Z_{t+h} for $|h| \geq 1$. This does not imply that, for example, $E[Z_t Z_{t+h} \mid Z_t] = 0$. After all, the whole point of the GARCH process is to allow such quantities to be nontrivial.

2.4 Parameter estimation for ARMA models

There are several ways to construct the likelihood function $L(\boldsymbol{\theta} \mid x_1, \ldots, x_n)$ for an ARMA model with $\boldsymbol{\theta} = (\phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q, \sigma^2)$. This section considers two alternatives: An easier (but less precise) way and a harder (but exact) way.

The **residuals** (often called innovations),

$$\epsilon_t := x_t - \phi_1 x_{t-1} - \cdots - \phi_p x_p - \theta_1 \epsilon_{t-1} - \cdots - \theta_q \epsilon_{t-q},$$

for $t = q+1, \ldots, n$, can be calculated from the observations. To do the calculation, $\epsilon_1, \ldots, \epsilon_q$ have to be assumed, and are usually set to zero, which is their unconditional expected value. The residuals serve as estimates for the $Z_{q+1}, \ldots, Z_n$, which are not observable. Let

$$\xi = (x_1, \ldots, x_p, \epsilon_1, \ldots, \epsilon_q).$$

The **conditional likelihood function** for an ARMA(p, q) model is derived by conditioning the density function on the first p observations and the first q residuals.

$$\begin{aligned} L(\boldsymbol{\theta} \mid \xi, x_{p+1}, \ldots, x_n) &= f(x_{p+1}, \ldots, x_n \mid \boldsymbol{\theta}, \xi) \\ &= f(x_{p+2}, \ldots, x_n \mid \boldsymbol{\theta}, \xi, x_{p+1}) f(x_{p+1} \mid \boldsymbol{\theta}, \xi) \\ &\vdots \\ &= f(x_n \mid \boldsymbol{\theta}, \xi, x_{p+1}, \ldots, x_n) \cdots f(x_{p+1} \mid \boldsymbol{\theta}, \xi). \end{aligned}$$

Assuming $\{Z_t\}$ Gaussian, all these terms are univariate Gaussian CDFs with mean zero (by an earlier assumption of ARMA processes) and variance σ^2. The conditional log likelihood can then be expressed as

$$l(\boldsymbol{\theta} \mid \xi, x_{p+1}, \ldots, x_n) = log(L(\boldsymbol{\theta} \mid \xi, x_{p+1}, \ldots, x_n)) = \frac{n-p-1}{2} log(2\pi) - \frac{n-p-1}{2} log(\sigma^2) - \sum_{t=p+1}^{n} \frac{\epsilon_t^2}{2\sigma^2}.$$

This expression can be estimated numerically w.r.t. $\boldsymbol{\theta}$ to obtain the maximum likelihood estimates (Hamilton, 1994).

Remark: The conditional likelihood is a good measure of goodness of fit even for non-Gaussian time series (Brockwell et al., 2016, p. 140).

Assuming $Z_t \sim$ GARCH$(1, 1)$ and $e_t \sim N(0, 1)$ gives the following distribution of X_t:

$$X_t \mid \xi, \sigma_t^2 \sim N(0, \sigma_t^2). \tag{8}$$

Let $\boldsymbol{\theta} = (\phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q, \alpha_0, \alpha_1, \beta_1)$ and let $\xi = (x_1, \ldots, x_p, \epsilon_1, \ldots, \epsilon_m)$, with $m = max(p, q)$. Given ξ and σ_p^2, the recursions

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1} + \beta_1 \sigma_{t-1}^2 \tag{9}$$

give σ_t for $t = p+1, \ldots n$, and the likelihood of the above model can be expressed in the following way:

$$
\begin{aligned}
L(\boldsymbol{\theta} \mid \xi, \sigma_p^2, x_{p+1}, \ldots, x_n) &= f(x_{p+1}, \ldots, x_n \mid \boldsymbol{\theta}, \xi, \sigma_p^2) \\
&= f(x_1, \ldots, x_n \mid \xi, \boldsymbol{\theta}\sigma_p^2) \\
&= \prod_{t=p+1}^{n} \frac{1}{\sigma_t \sqrt{2\pi}} e^{-\frac{\epsilon_t^2}{2\sigma_t^2}},
\end{aligned}
$$

Natural choices for $\epsilon_1, \ldots, \epsilon_m$ and σ_p^2 are their unconditional expected values, i.e., zero and $\frac{\alpha_0}{1-\alpha_1-\beta_1}$, respectively. The log of the above likelihood can be maximized numerically w.r.t. $\boldsymbol{\theta}$. Another popular choice of distribution of e_t is Student's t-distribution. When Matlab estimates the parameters of an ARMA model, it maximizes the above likelihoods (The MathWorks, 2022a). However, by default, the conditional values are $\xi = x_0, \ldots, x_{1-p}, \epsilon_0, \ldots, \epsilon_{1-q}$, which are all set to zero. To obtain confidence intervals for the estimates, Matlab uses a method called the **outer product of gradients** for ARMA models (The MathWorks, 2022c).

While the conditional likelihoods where easy to derive, one must make assumptions on the first residuals and the first p observations have to be used as presamples, leading to a slight reduction in the sample size. This should not be a problem if the sample size is large, as the first couple of terms only have a small contribution to the total log likelihood. It is, however, possible to calculate an **exact likelihood function** for an ARMA(p, q) model. Let Γ be the covariance matrix for $X_1, \ldots, X_n$, defined by $\Gamma_{ij} = cov(X_i, X_j) = E(X_i X_j)$, for $\{X_t\} \sim$ ARMA(p, q). Let $x = (x_1, \ldots, x_n)^T$ be a vector containing the n zero-mean observations of $\{X_t\}$ and let $\theta = (\phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q, \sigma^2)$. Assuming $Z_t \sim \mathrm{N}(0, \sigma^2)$, the exact likelihood is

$$
L(\boldsymbol{\theta} \mid x_1, \ldots, x_n) = (2\pi)^{-\frac{n}{2}} \det(\Gamma)^{-\frac{1}{2}} \exp\left(-\frac{1}{2} x^T \Gamma^{-1} x\right)
$$

If n is large, it can be too computer intensive to find Γ^{-1}. Additionally, the determinant of Γ may explode or vanish, leading to numerical problems. For these reasons, some more work is needed to derive a easily computable exact likelihood. One clever way to do this is through a **lower unit triangular decomposition** (LUTD) of Γ:

$$
\Gamma = ADA^T,
$$

where A is a lower triangular matrix with ones on the diagonal and D is a diagonal matrix.

Remark: Since Γ is a covariance matrix, it is also positive semi definite. It can be shown that all real positive semi definite matrices can be decomposed by the Cholesky decomposition $\Gamma = BB^T$, where B is a lower triangular matrix (not necessarily with ones on the diagonal) of non-negative numbers (Higham, 2009). Let $B = AD^{\frac{1}{2}}$, for A and D as defined above. Then it is easy to verify that the Cholesky decomposition is equivalent to the lower unit triangular decomposition. Thus, all covariance matrices can be decomposed in the above fashion.

For an ARMA(p, q) process, the LUTD simplifies the calculation of Γ^{-1} (defined as the inverse of all the elements on its diagonal) and $\det(\Gamma)$, since $\Gamma_{i1} = 0$ for $i > q + 1$. This implies that $A_{i,1} = 0$ for $i > q + 1$. The other columns of A has similar characteristics, resulting in the following matrix:

$$
A = \begin{bmatrix}
1 & 0 & 0 & \cdots & 0 & 0 \\
a_{21} & 1 & 0 & \cdots & 0 & 0 \\
a_{31} & a_{32} & 1 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \ldots & \vdots & \vdots \\
a_{q+1,1} & a_{q+1,2} & a_{q+1,3} & \cdots & 0 & 0 \\
0 & a_{q+2,2} & a_{q+2,3} & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \ldots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & a_{n,n-1} & 1
\end{bmatrix}. \tag{10}
$$

Note that

$$\det(\Gamma) = \det(ADA^T) = \det(ADA^T) = \det(A)\det(D)\det(A^T) = \det(A)^2\det(D) = \det(D)$$

and that $(AA^T) = I_n$, where I_n is the $n \times n$ identity matrix. Thus, the likelihood can be rewritten as

$$L(\boldsymbol{\theta} \mid x_1, \ldots, x_n) = (2\pi)^{-\frac{n}{2}}\det(D)^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(Ax)^T D^{-1} Ax\right)$$

Since D is a diagonal matrix, this is much simpler to maximize than the former expression of the likelihood. There are several ways to construct A and D. It can, for example, be done via the innovations algorithm, described in (Brockwell et al., 2016, p. 87). It is not clear how to do this exactly if $Z_t \sim \text{GARCH}(p, q)$. The series $\{\sigma_t^2\}$ has to start somewhere, but conditioning on an early iteration of σ_t^2 defeats the purpose of an exact likelihood. It might be possible to utilize the week stationarity property of $\{Z_t\}$ in a similar fashion as what is done with $\{X_t\}$. This is outside the scope of this thesis.

2.5 Prediction with ARMA models

The previous section shows how to obtain good estimates for the parameters in the ARMA model. In this section, it is assumed that the parameters of the ARMA model are known with certainty. Otherwise, the uncertainty of the estimates has to be accounted for when calculating the prediction error.

The goal of time series analysis is often to find the best possible prediction of $X_1, \ldots, X_{n+h}$ based on the observations $x_1, x_2, \ldots, x_n$. This requires more concrete notation. Let $P_nY = f(X_n, X_{n-1}, \ldots)$ be the prediction operator. It is the function that minimizes the **mean squared error** (MSE)

$$E[(Y - P_nY \mid X_n, X_{n-1}, \ldots)^2].$$

It can be shown that $P_nY = E[X_{n+1} \mid X_n, X_{n-1}, \ldots]$ minimizes the MSE (Brockwell et al., 2016, p. 34). Ideally, the task should be to find this function. In practice, however, this task is generally too difficult, and the objective is instead to find

$$\tilde{P}_nY = \hat{E}[X_{n+1} \mid X_n, X_{n-1}, \ldots],$$

the best linear predictor of Y based on $X_n, X_{n-1}, \ldots$. Thus, $\tilde{P}_nY$ is on the form $\sum_{i=1}^{n} c_i X_{n-i}$. Note that if Z_t is indeed Gaussian, $\tilde{P}_n = P_n$. The innovations algorithm gives a way to find $\tilde{P}_n X_{t+h}$ exactly (Brockwell et al., 2016, p. 87). It is unclear how this algorithm works if the noise is assumed to be $\text{GARCH}(p, q)$. In this thesis, the programming language Matlab was used to generate predictions from the ARMA models. Matlab uses a **large sample prediction procedure** to predict with ARMA models (The MathWorks, 2022b). For a causal and invertible ARMA process, the following identities hold:

$$X_{n+h} = \sum_{i=0}^{\infty} \psi_i Z_{n+h-i}$$

$$Z_{n+h} = X_{n+h} + \sum_{i=1}^{\infty} \pi_i Z_{n+h-i}$$

Applying $\tilde{P}_n$ on both sides of both equations gives

$$\tilde{P}_n X_{n+h} = \sum_{i=h}^{\infty} \psi_i Z_{n+h-i} \tag{11}$$

$$\tilde{P}_n X_{n+h} = -\sum_{i=1}^{\infty} \pi_i \tilde{P}_n X_{n+h-i}, \tag{12}$$

since $\tilde{P}_n Z_{n+h-i} = 0$, for $i \leq h$, and $\tilde{P}_n Z_{n+h-i} = Z_{n+h-i}$, for $i > h$. Note that $\tilde{P}_n X_{n+h-i} = X_{n+h-i}$ for $i > h$. Since $X_1, \ldots, X_n$ are observed, Equation (12) defines a recursive way to obtain $\tilde{P}_n X_{t+h}$. Given the infinite past of X_n, Equation (12) gives an exact prediction of X_{n+h}. However, good approximations can be made by truncating the sum after n observations when the coefficients $|\pi_i|$ are small and n is large. The above equations give a neat expression of $X_{n+h} - \tilde{P}_n X_{n+h}$, which is the first step towards calculating the MSE of the forecast:

$$X_{n+h} - \tilde{P}_n X_{n+h} = \sum_{i=0}^{h-1} \psi_i Z_{n+h-i}.$$

For an ARMA process with $Z_t \sim WN(0, \sigma^2)$, the MSE of the forecast after h steps is

$$E[(X_{n+h} - \tilde{P}_n X_{n+h})^2 \mid X_n, X_{n-1}, \ldots] = \sigma^2 \sum_{i=0}^{h-1} \psi_i^2. \tag{13}$$

If the coefficients $|\psi_i|$ are small, this quickly converges when h is large enough. When assuming $Z_t \sim \text{GARCH}(1,1)$, Equation (12) still holds. However, the MSE changes in the following way:

$$E[(X_{n+h} - \tilde{P}_n X_{n+h})^2 \mid X_n, X_{n-1}, \ldots] = \sigma_t^2 \sum_{i=0}^{h-1} \psi_i^2 = (\alpha_0 + \alpha_1 \epsilon_{t-1} + \beta \sigma_{t-1}^2) \sum_{i=0}^{h-1} \psi_i^2$$

where σ_t^2 can be calculated by the recursions described in Equation (9). Matlab uses Equation (12) to generate predictions and use Equation (13) (or Equation (2.5) with GARCH noise) to generate the associated prediction error (The MathWorks, 2022b).

2.6 Simulation with ARMA models

Simulation can be used to study the properties of a specific ARMA(p, q) process

$$X_t - \phi_1 X_{t-1} - \ldots \phi_p X_{t-p} = Z_t - \theta_1 Z_{t-1} - \ldots \theta_q Z_{t-q}$$

with known parameters $\phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q, \sigma^2$. Assume that the task is to simulate a realization $\tilde{X}_{n+1}, \ldots, \tilde{X}_{n+h}$ from an ARMA(p, q) based on the observations $x_1, \ldots, x_n$. The residuals, ϵ_t, for $q < t \leq n$, function as estimators for the unobserved $Z_{q+1}, \ldots, Z_n$, assuming $Z_t = 0$, for $t \leq q$. This is not a wild assumption as it is their unconditional expectation (with or without GARCH noise). Simulating from X_{n+1} reduces to simulating from Z_{n+1}. In principle, no assumption of the particular distribution of Z_t needs to be assumed for an ARMA process. There are at least two ways to deal with this. The first way is by bootstrapping the already observed residuals $\epsilon_{q+1}, \ldots, \epsilon_n$. The second method is to assume a distribution on Z_t. The most popular choices are the Gaussian distribution and Student's t-distribution. These distributions can easily be simulated from. The latter simulation method, with an assumed normal distribution, was used to simulate ARMA processes in this thesis, as this is the default approach in Matlab (The MathWorks, 2022d). Once Z_{n+1} is simulated, the simulated value of X_{n+1} easily follows from the ARMA equations given the observations and the residuals. Repeating this procedure for Z_{n+2}, gives a simulated realization of X_{n+2}, and so on. Thus, it is possible to simulate as far as necessary into the future. If the noise is assumed to be a GARCH(p, q) process, Z_{n+1} is simulated from the GARCH process, where the residuals $\epsilon_{q+1}, \ldots, \epsilon_n$ are treated similarly as the observations where above. This time, a particular distribution of e_t has to be assumed, where the default, Gaussian distribution was chosen.

2.7 Model Diagnostics for ARMA models

It is useful to qualify how good a certain ARMA model is fitted to the data. Most notably, this makes it possible to compare different model configurations against each other, to find the best model for the data.

To quantify how well the model fits the observed time series data, the **Akaike information criterion** (AIC) can be used. Denote the maximum likelihood estimates as $\hat{\theta}$. The AIC for a model is defined as

$$AIC(\hat{\theta}) = -2l(\hat{\theta}) + 2k,$$

where k is the number of parameters fitted. As an example, for a SARIMA$(p, d, q) \times (P, D, Q)_s$ process, $k = p + q + P + Q$. The **corrected AIC** (AICc) is a bias-corrected version of the AIC and is defined as

$$AICc(\hat{\theta}) = AIC(\hat{\theta}) + \frac{2k(k+1)}{n-k-1},$$

where n is the sample size used to fit the model. In the case of small sample size, the AICc is usually recommended, as this is when the bias is most prevalent. Note that the AICc converges to the AIC when a large sample size is available. The AICc represents the information lost by the model compared to the actual data. Thus, a lower value indicates a better fit.

The **auto-correlation function** (ACF) for a time series $\{X_t\}$ is defined as

$$\alpha(h) = Cor(X_t, X_{t+h})$$

A plot of the ACF is a nice way to visualize the dependencies of a time series and to choose the order of q for an ARMA model. The values of h are displayed on the x-axis and are called lags. Each lag has the associated value $\alpha(h)$ on the y-axis. 95% confidence bands are also included in the plot. Thus, if for example, the value at lag 3 is significant, it may be wise to fit an ARMA model with $q \geq 3$ to account for this. In this thesis, the order of the models where already decided, and the ACF plot was only used to assess the fit of the SARIMA models. This was done by plotting the ACF of the residuals $\{\epsilon_t\}$ of fitted models.

The residuals of a time series model should ideally be independent, as this means that all the dependencies in the data are explained by the model. By looking at the ACF plot, one can assess whether the residuals are uncorrelated. In this case, the lags of the ACF plot should not be significantly different from zero. However, uncorrelated residuals do not imply independence. If, in particular, the squares of the residuals show significant lags, the residuals still exhibit some serial dependence. Applying a GARCH model to the ARMA model might solve this problem, as it models the dependence between the variances of $\{Z_t\}$.

Engel's ARCH test is another way examine if the residuals exhibit conditional heteroscedasticity. Let $\phi(B)X_t = \theta(B)Z_t$, for $Z_t = \sigma_t^2 e_t$, with $e_t \sim IID(0, 1)$. Then

$$Var(X_t \mid X_{t-1}, X_{t-2}, \ldots) = Var(Z_t \mid X_{t-1}, X_{t-2}, \ldots) = E[Z_t^2 \mid X_{t-1}, X_{t-2}, \ldots] = \sigma_t^2,$$

since $E[Z_t Z_{t+h}] = 0$. Thus, to identify heteroscedasticity, one only need to look at the auto-correlation process. Even if the model accounts for all the correlation of $\{X_t\}$, the variables in $\{\epsilon_t\}$ might still be dependent. By expressing the squared residuals as a AR(m) process

$$\epsilon_t^2 = \alpha_1^2 \epsilon_{t-1}^2 + \cdots + \alpha_m^2 \epsilon_{t-m}^2 + w_t, \quad w_t \sim WN(0, 1), \tag{14}$$

it is clear that homoscedasticity is obtained only when $\alpha_0 = \cdots, \alpha_m = 0$. This is exactly what is done in Engel's ARCH test, where the null hypothesis is

$$H_0 : \alpha_0 = \cdots, \alpha_m = 0$$

for α_j as in Equation (14), while the alternative hypothesis is

$$H_a : \epsilon_t^2 = \alpha_1^2 \epsilon_{t-1}^2 + \cdots + \alpha_m^2 \epsilon_{t-m}^2 + w_t,$$

with $\alpha_j \neq 0$ for at least one $j = 1, \ldots, m$. Under H_0, it can be shown that $\epsilon_t^2 \sim \chi_m^2$ (Engle, 1982). In Matlab, the amount of significant lags to test for (given by m in Equation (14)) can be specified (The MathWorks, 2022e). The ARCH test can also be used to test for GARCH(p, q) processes, since it is locally equivalent to an ARCH$(p + q)$ process.

2.8 The Machine Learning Model

Machine learning models usually have a large number of parameters, which in principle gives them the flexibility to model any function. Machine learning models have recently shown good results in the field of time series forecasting, and have become a contender to the more traditional statistical time series models (Kiranyaz et al., 2021). This section covers two types of machine learning models, that will later be combined and used to predict future daily new cases for the two Covid-19 data sets. It should be noted that the details of each operation described in this section can be modified by the user. This section merely describes how the methods where implemented in this thesis, using an API called TensorFlow.

2.8.1 Convolutional Neural Networks

Convolutional neural networks (CNN) have increased in popularity over the last decades. Their first big achievements where in the field of image classification (Kiranyaz et al., 2021). Image data usually have lots and lots of features (one for each pixel). Using traditional fully connected networks to do classification based on all these features is time consuming, as too many weights must be tuned, and usually results in overfitting. The CNNs solves these problems by having sparse connections between the first layers of neurons.

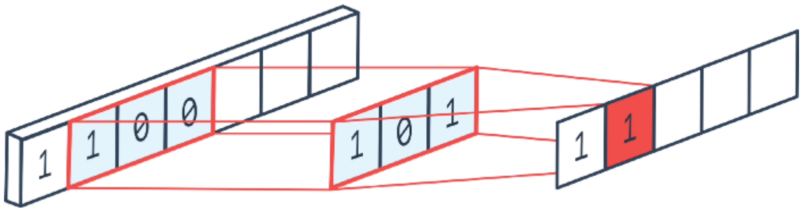


Figure 2: An illustration of a one-dimensional kernel operation with kernel size is 3 (TowardsDataScience, n.d.).

Recently, CNNs has shown great strides on one-dimensional data, e.g., time series data (Kiranyaz et al., 2021). The key to modeling time series data is to capture the dependence between each time step. CNNs does exactly this, since they create new features by convolving neighbouring data points together, as illustrated by Figure 2. With TensorFlow in the programming language Python, there are several settings specifying the exact nature of the convolutional operation. The following definition only addresses the operation with default settings, as this is what will be used later (Wei Zhang et al., 2017):

Definition 2.13 (Convolution in one-dimension)**.** *Let $x = (x_1, \dots, x_n)$ be the observed time series and let K be a $s \times 1$ convolutional kernel with corresponding bias b. Define $Y_i = (x_i, \dots, x_{i+s})$, for $i = 1, \dots, n-s-1$. The one-dimensional convolution operation can be defined in the following way:*

$$
\begin{aligned}
F_1 &= f(Y_1 K + b) \\
F_2 &= f(Y_2 K + b) \\
&\vdots \\
F_{n-s+1} &= f(Y_{n-s+1} K + b),
\end{aligned}
$$

where $f(\cdot)$ is called an activation function and $F = (F_1, \dots, F_{n-s+1})$ is the convolved time series, often called a feature map.

The activation function is usually non-linear. If not, it would be impossible for the CNN to fit a non-linear function to the data, as the convolution operation is linear. Recently, because of its efficiency, the most popular activation function has been the rectified linear unit (ReLu), i.e., the function $f(x) = max(x, 0)$. However, other functions have their use cases. The activation function completes a convolutional layer.

A convolutional network consists of one or more convolutional layers, exemplified in Figure 3. Each layer may have multiple filters and each filter is defined by its associated kernel. The convolutional operation from Definition 2.13 happens in the filters, where each filter gives a mapping to the next layer of the network. This results in sparse connections between the layers of the network, which may reduce overfitting. Additionally, the feature maps are in some sense shift-invariant, since each element is a function all neighbouring points of the input time series.

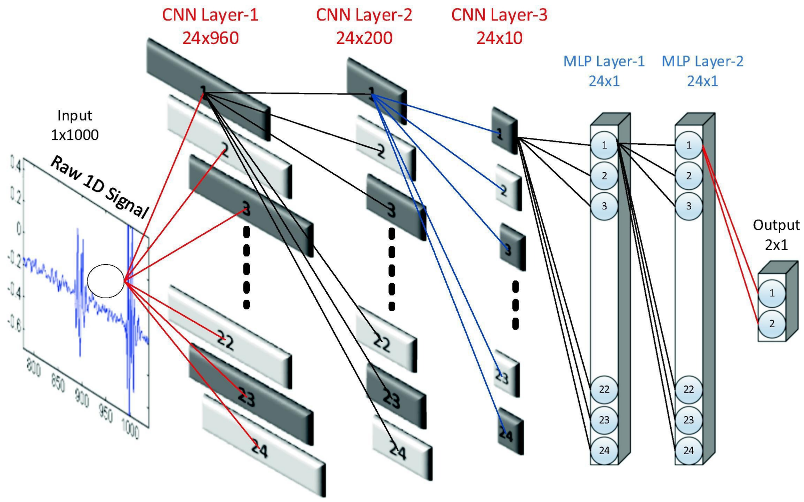


Figure 3: An illustration of a one-dimensional convolutional neural network (Kiranyaz et al., 2021).

The elements inside a kernel are called weights and are tuned through backpropagation. The goal of backpropagation is to minimize a loss function by tuning the weights in the opposite way of its gradient (Z. Zhang, 2016). The choice of loss function depends on the problem.

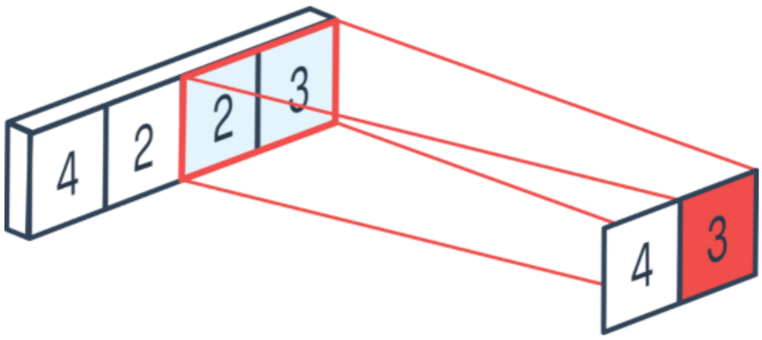
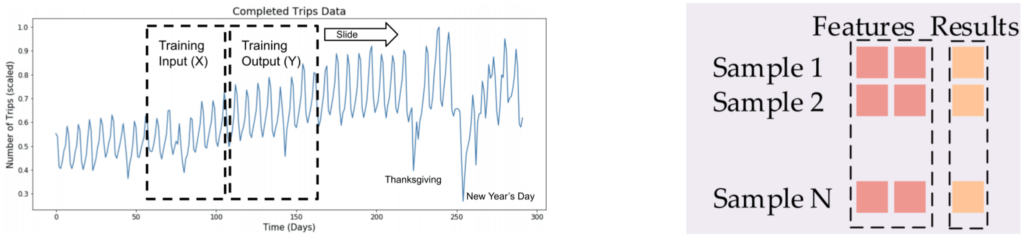


Figure 4: An illustration of a one-dimensional pooling operation, with size 2 and stride 2 (Peltarion, 2022).

The feature maps vary from filter to filter. To reduce the variation between the feature maps, it is common to add a pooling layer after a convolutional layer. They process the incoming feature map in a similar fashion as the convolutional layers. However, instead of applying a kernel section wise, it computes a summarizing statistic over each section, as shown in Figure 4. The dropout layer has a similar purpose. At a specified rate r, the neurons are set to zero, while the remaining neurons are multiplied by $\frac{1}{1-r}$, to keep the sum of the neurons unchanged (TensorFlow, 2022a). This is a regularization technique that forces the model to consider more features when training.



(a) A visualization of how samples are construction for time series data (MachineLearningMastery, n.d.).

(b) The samples are stored in two matrices, the input matrix and the output matrix (J. Zhang et al., 2021)

Figure 5: Construction of samples from time series data.

To tune the weights of a one-dimensional CNN, the input series has to be reformatted into samples of input-output pairs, as Figure 5 illustrates. The input matrix contains subsequences of the input time series of the same length in each row, and the output matrix contains the ensuing time point(s). For example, if the task is to use the previous n days to predict the next m days, then each row of the input matrix should contain n elements, while the output matrix should contain the next m elements of the input time series. This gives the network input-output samples to learn from and can perform the task at hand once it is trained. Each filter is applied row wise to the input matrix to produce the feature maps. The predictions from the network (one for each row) can then be compared to the respective row in the output matrix, and the weighs can be tuned accordingly.

The output of the CNN can then be used as input in a different model, that can interpret the newly processed data. A popular choice in time series modeling is to apply a Long-Short Term Memory (LSTM) model to the convolved data (Luan et al., 2019).

2.8.2 Long Short Term Memory models

The LSTM model is a modified version of a Recurrent Neural Network (RNN) and is a popular option for time series forecasting. A traditional RNN can be seen in Figure 6. The model takes in the input matrix $x = (x_1, \dots, x_n)$ and returns the output matrix $o = (o_1, \dots, o_n)$. Note that x is not the observed time series. For one-dimensional data, x could either be the input matrix created from the transformed data, with each x_i being a sub sequence of observed data (as described in Figure 5), or it could be the output of the previous layer (for example a CNN). The neurons are connected across time steps through the internal state vectors $s = (s_1, \dots, s_n)$. The mathematical description of a RNN is the following (W. Zhang et al., 2019):

$$\begin{aligned} s_t &= g_h \left(U x_t + W s_{t-1} + b_U + b_W \right) \\ o_t &= g_o \left(V s_t + b_V \right), \end{aligned} \tag{15}$$

where U, V and W are weight matrices and g_h and g_o are activation functions, corresponding to the hidden and output layer, respectively. Excluded from Figure 6 are all the bias vectors, b_U, b_V and b_W corresponding to U, V and W, respectively.

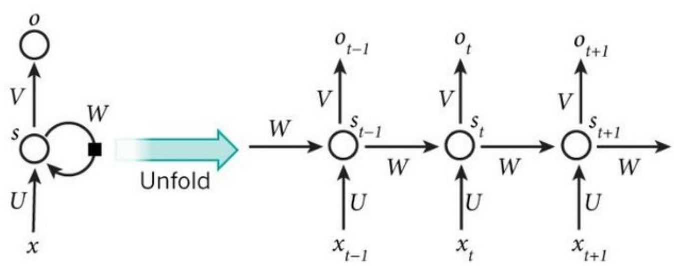


Figure 6: An illustration of a RNN (W. Zhang et al., 2019).

Just as for CNNs, all weight matrices are trained by backpropagation. However, for RNNs, the backpropagation algorithm also has to traverse all time steps $t = 1, \dots n$. This is no problem for short sequences of data, however, for longer sequences, the gradient comprises of more and more multiplicative factors. Consider the backpropagation of the weights in W. Let $\mathcal{L}_t := \mathcal{L}(x_t - o_t)$ be the contribution to the loss function at time step t. The loss function is the sum of all its contributions $L := \sum_{t=1}^{n} \mathcal{L}_t$. Since this is purely a demonstration, the basic gradient decent algorithm was used to calculate the gradient (Ruder, 2016). The error gradient w.r.t. W is (Pascanu et al., 2012):

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{n} \frac{\partial \mathcal{L}_t}{\partial W}.$$

The gradient of the contribution to the loss at time step t, $\frac{\partial \mathcal{L}_t}{\partial W}$, is propagated back to each time step:

$$\begin{aligned}
\frac{\partial \mathcal{L}_t}{\partial W} &= \underbrace{\frac{\partial \mathcal{L}_t}{\partial o_t}\frac{\partial o_t}{\partial s_t}\frac{\partial s_t}{\partial W}}_{\substack{\text{contribution to the loss at time } t \\ \text{from internal state } s_t}} + \dots + \underbrace{\frac{\partial \mathcal{L}_t}{\partial o_t}\frac{\partial o_t}{\partial s_t}\frac{\partial s_t}{\partial s_{t-1}}\frac{\partial s_{t-1}}{\partial s_{t-2}}\cdots\frac{\partial s_1}{\partial W}}_{\substack{\text{contribution to the loss at time } t \\ \text{from internal state } s_1}} \\
&= \frac{\partial \mathcal{L}_t}{\partial o_t}\frac{\partial o_t}{\partial s_t}\frac{\partial s_t}{\partial W} + \sum_{k=2}^{t} \frac{\partial \mathcal{L}_t}{\partial o_k}\frac{\partial o_k}{\partial s_k}\frac{\partial s_k}{\partial s_{k-1}} \prod_{i=0}^{t-k-1} \frac{\partial s_{t-i}}{\partial s_{t-i-1}}\frac{\partial s_k}{\partial W}.
\end{aligned}$$

Thus, the total loss from n observations is

$$\begin{aligned}
\frac{\partial L}{\partial W} &= \sum_{t=1}^{n}\sum_{k=1}^{t} \frac{\partial \mathcal{L}_k}{\partial o_k}\frac{\partial o_k}{\partial s_k}\frac{\partial s_k}{\partial s_{k-1}} \prod_{i=0}^{t-k-1} \frac{\partial s_{t-i}}{\partial s_{t-i-1}}\frac{\partial s_k}{\partial W} \\
&= \sum_{t=1}^{n}\sum_{k=1}^{t} \frac{\partial \mathcal{L}_k}{\partial o_k}\frac{\partial o_k}{\partial s_k}\frac{\partial s_k}{\partial s_{k-1}} \prod_{i=0}^{t-k-1} \left[\frac{\partial}{\partial s_{t-i-1}} g_h(Ux_t + Ws_{t-i-1} + b_U + b_W)\right]\frac{\partial s_k}{\partial W} \\
&= \sum_{t=1}^{n}\sum_{k=1}^{t} \frac{\partial \mathcal{L}_k}{\partial o_k}\frac{\partial o_k}{\partial s_k}\frac{\partial s_k}{\partial s_{k-1}} W^{t-k} \prod_{i=0}^{t-k-1} g_h'(Ux_t + Ws_{t-i-1} + b_U + b_W)\frac{\partial s_k}{\partial W}.
\end{aligned}$$

The long term contributions to the gradient suffer from the growing exponent in the weight matrix W. If not all weights are equal to unity, this effectively results in a vanishing or an exploding contribution to the full gradient. The same problem occurs when backpropagating U and the biases b_W and b_U. This illustrates why the traditional RNN struggles to learn long term dependencies. This is known as the vanishing or exploding gradient problem.

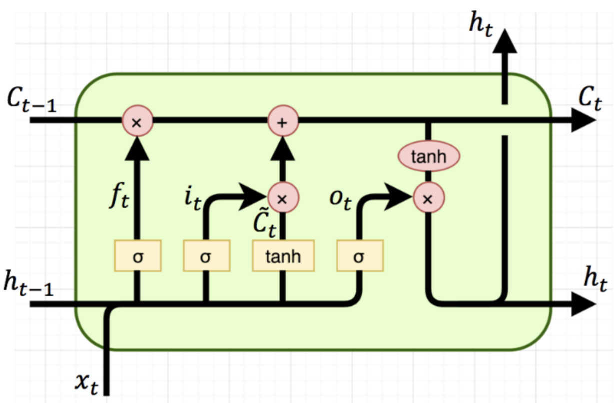


Figure 7: An illustration of a LSTM cell (Varsamopoulos et al., 2018).

A remedy to this problem is the LSTM model, which is illustrated in Figure 7. It uses memory cells to decide what information to forget from the previous cells, and what new information to keep. This information is then sent through to the internal state, just as for traditional RNNs. Let $x = (x_1, \ldots, x_n)$ be the same input matrix as described earlier. The following set of equations define the LSTM model (Varsamopoulos et al., 2018):

$$\begin{aligned}
f_t &= g_1\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \\
i_t &= g_1\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\
o_t &= g_1\left(W_o\, [h_{t-1}, x_t] + b_o\right) \\
\tilde{C}_t &= g_2\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
h_t &= o_t * g_2\left(C_t\right),
\end{aligned} \tag{16}$$

where g_1 and g_2 are activation functions.

In the above equation, the forget gate f_t controls what information from C_{t-1} to "forget" based on the new information from h_{t-1} and x_t. The input gate at time t, called i_t, specifies what new information should be passed through to the cell state C_t at the current time step. The output gate o_t controls what information from C_t to output at time step t. The output vector is denoted h_t. It is used as input in the cell corresponding to the next time step, $t + 1$.

The vectors C_t, C_{t-1}, o_t and h_t have the same length, often referred to as the number of *units* u of the LSTM-layer. The number of units has to be specified during implementation. Similar to other machine learning models, the weight matrices W_* and the bias vectors b_* are randomly initialized and are trained through **back propagation in time**.

The construction of the LSTM-cell results in an additive structure of the factors of the gradient. Let $L = \sum_{t=1}^{n} \mathcal{L}_t$ be the loss function. To show that the LSTM model solve the vanishing and exploding gradient problem, it has to be shown that all partial gradients manage the problem. The derivation of the gradient for W_f, W_i and W_C and the corresponding biases are all similar to that of W in a RNN. For W_f, the gradient is

$$\frac{\partial L}{\partial W_f} = \sum_{t=1}^{n} \sum_{k=1}^{t} \frac{\partial \mathcal{L}_k}{\partial h_k} \frac{\partial h_k}{\partial C_k} \frac{\partial C_k}{\partial C_{k-1}} \prod_{i=0}^{t-k-1} \frac{\partial C_{t-i}}{\partial C_{t-i-1}} \frac{\partial C_k}{\partial f_k} \frac{\partial f_k}{\partial W_f}.$$

However, the terms $\frac{\partial C_{t-i}}{\partial C_{t-i-1}}$ from the LSTM model behaves nicer than $\frac{\partial s_{t-i}}{\partial s_{t-i-1}}$ from the RNN:

$$\frac{\partial C_{t-i}}{\partial C_{t-i-1}} = \frac{\partial f_{t-i}}{\partial C_{t-i-1}} C_{t-i-1} + \frac{\partial C_{t-i-1}}{\partial C_{t-i-1}} f_{t-i} + \frac{\partial i_{t-i}}{\partial C_{t-i-1}} \tilde{C}_{t-i} + \frac{\partial \tilde{C}_{t-i}}{\partial C_{t-i-1}} i_{t-i},$$

where

$$\begin{aligned}
\frac{\partial f_{t-i}}{\partial C_{t-i-1}} &= W_f \frac{\partial h_{t-i-1}}{\partial C_{t-i-1}} g_1'(W_f \cdot [h_{t-i-1}, x_{t-i}] + b_f) \\
&= W_f \cdot o_{t-i-1} \cdot C_{t-i-1} \cdot g_2'(C_{t-i-1}) \cdot g_1'(W_f \cdot [h_{t-i-1}, x_{t-i}] + b_f),
\end{aligned}$$

$$\begin{aligned}
\frac{\partial i_{t-i}}{\partial C_{t-i-1}} &= W_i \frac{\partial h_{t-i-1}}{\partial C_{t-i-1}} g_1'(W_i \cdot [h_{t-i-1}, x_{t-i}] + b_i) \\
&= W_i \cdot o_{t-i-1} \cdot C_{t-i-1} \cdot g_2'(C_{t-i-1}) \cdot g_1'(W_i \cdot [h_{t-i-1}, x_{t-i}] + b_i),
\end{aligned}$$

and

$$\begin{aligned}\frac{\partial \tilde{C}_{t-i}}{\partial C_{t-i-1}} &= W_C \frac{\partial h_{t-i-1}}{\partial C_{t-i-1}} g_1'(W_C \cdot [h_{t-i-1}, x_{t-i}] + b_C) \\ &= W_C \cdot o_{t-i-1} \cdot C_{t-i-1} \cdot g_2'(C_{t-i-1}) \cdot g_1'(W_C \cdot [h_{t-i-1}, x_{t-i}] + b_C).\end{aligned}$$

Hence

$$\begin{aligned}\frac{\partial C_{t-i}}{\partial C_{t-i-1}} =& W_f \cdot o_{t-i-1} \cdot C_{t-i-1} \cdot g_2'(C_{t-i-1}) \cdot g_1'(W_f \cdot [h_{t-i-1}, x_{t-i}] + b_f) \cdot C_{t-i-1} \\ &+ f_{t-i} \\ &+ W_i \cdot o_{t-i-1} \cdot C_{t-i-1} \cdot g_2'(C_{t-i-1}) \cdot g_1'(W_i \cdot [h_{t-i-1}, x_{t-i}] + b_i) \cdot \tilde{C}_{t-i} \\ &+ W_C \cdot o_{t-i-1} \cdot C_{t-i-1} \cdot g_2'(C_{t-i-1}) \cdot g_1'(W_C \cdot [h_{t-i-1}, x_{t-i}] + b_C) \cdot i_{t-i}.\end{aligned}$$

Since f_t has direct influence on the value of $\frac{\partial C_{t-i}}{\partial C_{t-i-1}}$ and has a direct update mechanism, the model can better regulate the long term gradients. This enables the model to learn long term dependencies in the direction of W_f.

The weights W_i, W_C and the biases b_f, b_i and b_C can be calculated in a similar way:

$$\frac{\partial L}{\partial W_i} = \sum_{t=1}^{n} \sum_{k=1}^{t} \frac{\partial \mathcal{L}_k}{\partial h_k} \frac{\partial h_k}{\partial C_k} \frac{\partial C_k}{\partial C_{k-1}} \prod_{i=0}^{t-k-1} \frac{\partial C_{t-i}}{\partial C_{t-i-1}} \frac{\partial C_k}{\partial i_k} \frac{\partial i_k}{\partial W_i}.$$

$$\frac{\partial L}{\partial W_C} = \sum_{t=1}^{n} \sum_{k=1}^{t} \frac{\partial \mathcal{L}_k}{\partial h_k} \frac{\partial h_k}{\partial C_k} \frac{\partial C_k}{\partial C_{k-1}} \prod_{i=0}^{t-k-1} \frac{\partial C_{t-i}}{\partial C_{t-i-1}} \frac{\partial C_k}{\partial \tilde{C}_k} \frac{\partial \tilde{C}_k}{\partial W_C}.$$

$$\frac{\partial L}{\partial b_f} = \sum_{t=1}^{n} \sum_{k=1}^{t} \frac{\partial \mathcal{L}_k}{\partial h_k} \frac{\partial h_k}{\partial C_k} \frac{\partial C_k}{\partial C_{k-1}} \prod_{i=0}^{t-k-1} \frac{\partial C_{t-i}}{\partial C_{t-i-1}} \frac{\partial C_k}{\partial f_k} \frac{\partial f_k}{\partial b_f}.$$

$$\frac{\partial L}{\partial b_i} = \sum_{t=1}^{n} \sum_{k=1}^{t} \frac{\partial \mathcal{L}_k}{\partial h_k} \frac{\partial h_k}{\partial C_k} \frac{\partial C_k}{\partial C_{k-1}} \prod_{i=0}^{t-k-1} \frac{\partial C_{t-i}}{\partial C_{t-i-1}} \frac{\partial C_k}{\partial i_k} \frac{\partial i_k}{\partial b_i}.$$

$$\frac{\partial L}{\partial b_C} = \sum_{t=1}^{n} \sum_{k=1}^{t} \frac{\partial \mathcal{L}_k}{\partial h_k} \frac{\partial h_k}{\partial C_k} \frac{\partial C_k}{\partial C_{k-1}} \prod_{i=0}^{t-k-1} \frac{\partial C_{t-i}}{\partial C_{t-i-1}} \frac{\partial C_k}{\partial \tilde{C}_k} \frac{\partial \tilde{C}_k}{\partial b_C}.$$

In each of these error gradient, the long term memory is ensured by the forget gate. Finally, the gradients for the output gate are the following:

$$\frac{\partial L}{\partial W_o} = \sum_{t=1}^{n} \sum_{k=1}^{t} \frac{\partial \mathcal{L}_k}{\partial h_k} \frac{\partial h_k}{\partial o_k} \frac{\partial o_k}{\partial W_o} = \sum_{t=1}^{n} \sum_{k=1}^{t} \frac{\partial \mathcal{L}_k}{\partial h_k} g_2(C_k) \frac{\partial o_k}{\partial W_o}$$

$$\frac{\partial L}{\partial b_o} = \sum_{t=1}^{n} \sum_{k=1}^{t} \frac{\partial \mathcal{L}_k}{\partial h_k} \frac{\partial h_k}{\partial o_k} \frac{\partial o_k}{\partial W_o} = \sum_{t=1}^{n} \sum_{k=1}^{t} \frac{\partial \mathcal{L}_k}{\partial h_k} g_2(C_k) \frac{\partial o_k}{\partial b_o},$$

which is no problem since it does not contain the multiplicative term. This shows that, in principle, the gradient manages to learn long term dependencies in each direction.

The series of vectors $[h_1, \ldots, h_n]$ is the output of the model, but the default behaviour in TensorFlow (2022b) is to return only the last vector h_n, since this vector are associated with x_{n+1}. This is regulated by the input parameter `return_sequences`. It is important to remark that h_n is not supposed to be the prediction of X_{n+1}. Remember that h_t are vectors of length u. A fully connected layer is often used after the LSTM layer(s) to interpret h_n, and to return a single predicted value for each ensuing day. This forecast can then be compared to the observed values, to compute the model's accuracy.

2.9 Measures of accuracy

To compare the performance of each model, a measure of the accuracy of the forecasts from each model is needed. Two accuracy measures are considered in this thesis. The first measure is the **relative root mean squared error** (RRMSE). Given a n-day prediction $\hat{y}$ with corresponding test data y, the RRMSE is calculated by

$$\text{RRMSE}\,(\hat{y}, y) = \frac{\sqrt{(1/n)\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}}{(1/n)\sum_{i=1}^{n}|y_i|} \times 100\%$$

RRMSE has the advantage of being comparable between data sets of different scales. By dividing the root mean squared error by the total scale of the testing data, a data set with millions of new cases of Covid-19, like the global data set, can be compared with models for the Norwegian data set. According to Zain et al. (2021) and Li et al. (2013), the accuracy of a forecast is deemed excellent when the RRMSE is less than 10 percent, good when the RRMSE is between 10 and 20 percent, fair when the RRMSE is between 20 and 30 percent, and poor when the RRMSE is greater than 30 percent.

The **mean absolute percentage error** (MAPE) is calculated in the following way:

$$\text{MAPE}\,(\hat{y}, y) = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{|y_i|} \times 100\%$$

The MAPE is similar to the RRMSE in that it divides the error by the values in the observed test data. It can therefore, in principle, be compared across data sets with differing scales. The difference between the MAPE and the RRMSE is that the division happens inside the summation, and that the absolute value is used to measure the deviation between the prediction and the observed value. This means that extreme outliers do not contribute as much to the MAPE as it would when using the RRMSE, especially when accompanied with higher values in the test data.

3 Methods

This section specifies the three models used in this thesis and gives the necessary details to replicate the upcoming results.

3.1 Defining the three models

The previous section described general results for ARMA(p, q) models, CNNs and LSTM models. The three specific models used in this thesis are now introduced.

3.1.1 The SARIMA model

In the winter of 2020, Taraldsen (2020b) published a so-called toy model that fit the Covid-19 new-cases data set from Norway nicely. After transforming the new-cases time series $\{Y_t\}$:

$$X_t = \log(\max(Y_t, 0.1)), \quad \text{for } t = 1, \ldots, n,$$

he fitted the following model to $\{X_t\}$:

$$(1 - B)(1 - B^7)X_t = (1 - \theta_{MA}B)(1 - \Theta_{SMA}B^7)Z_t, \quad Z_t \sim N(0, \sigma^2), \tag{17}$$

i.e., a SARIMA$(0, 1, 1) \times (0, 1, 1)_7$ model. The model is completely defined by the three parameters θ_{MA}, Θ_{SMA} and σ^2. The model will later be referred to as **the SARIMA model**.

Remark: An equivalent formulation of the above SARIMA model is to model the differenced time series $\{\tilde{X}_t\} = (1 - B)(1 - B^7)X_t$ by the following MA(8) process:

$$\tilde{X}_t = (1 - \theta_{MA}B)(1 - \Theta_{SMA}B^7)Z_t. \tag{18}$$

A precondition for fitting the SARIMA model to $\{X_t\}$ is that the time series $\{\tilde{X}_t\}$ is stationary. This is explored in Section 4.2.

3.1.2 The Gandalf model

The SARIMA model fits surprisingly well to the Norwegian data set, but Taraldsen discovered that the squared residuals had a significant correlation at lag 1 (replicated in Figure 15a). To incorporate this into the model, he used the same SARIMA model, but with the assumption that $Z_t \sim$ GARCH$(1, 1)$. Section 2.3.1 established that $\alpha_1 + \beta_1 < 1$. Additionally, Taraldsen fixed $\alpha_0 = 0.001$, because of numeric problems when α_0 approached zero. The model is described by the following equations:

$$\begin{aligned} (1 - B)(1 - B^7)X_t &= (1 - \theta_{MA}B)(1 - \Theta_{SMA}B^7)Z_t, \\ Z_t = \sigma_t^2 e_t, \quad \sigma_t^2 &= 0.001 + \alpha_1 Z_{t-1}^2 + \beta_1 \sigma_{t-1}^2. \end{aligned}$$

This model is also a SARIMA model. However, to not confuse the reader, this model is called the **Gandalf model**. Sometimes it is useful to refer both the Gandalf model and the SARIMA model together. The plural **SARIMA models** will be used for this purpose.

3.1.3 The CNN-LSTM model

At the time of publishing (July 23rd 2021), a hybrid between a CNN and a LSTM model gave one of the best predictions of the global new-cases data set (Zain et al., 2021). An outline of the model can be seen in Figure 8. First observe that the model is designed to take sequences of new cases of seven consecutive days as input, and outputs the amount of new cases on the ensuing eight day. The input-output pairs are constructed accordingly. The seven-day sequences are passed through two one-dimensional convolutional layers with 64 filters each, with all filters having kernel size 3. Both convolutional layers use ReLU as activation function. After the convolutional layers, a pooling layer of size 1 is added, followed by a dropout layer. However, the dropout rate was not unspecified.

The remaining filters were then flattened into one long vector, which was used as input to the LSTM layer with 200 units. The default activation functions were used, i.e., $g_1(\cdot) = \text{sigmoid}(\cdot)$ and $g_2(\cdot) = \tanh(\cdot)$ (TensorFlow, 2022b). Finally, there are two fully connected layers; the number of units in the first of these layers was not specified and was set to 50, while the second one has 1 unit to be able to output one single value. The fully connected layers used their default activation functions, i.e., ReLU.

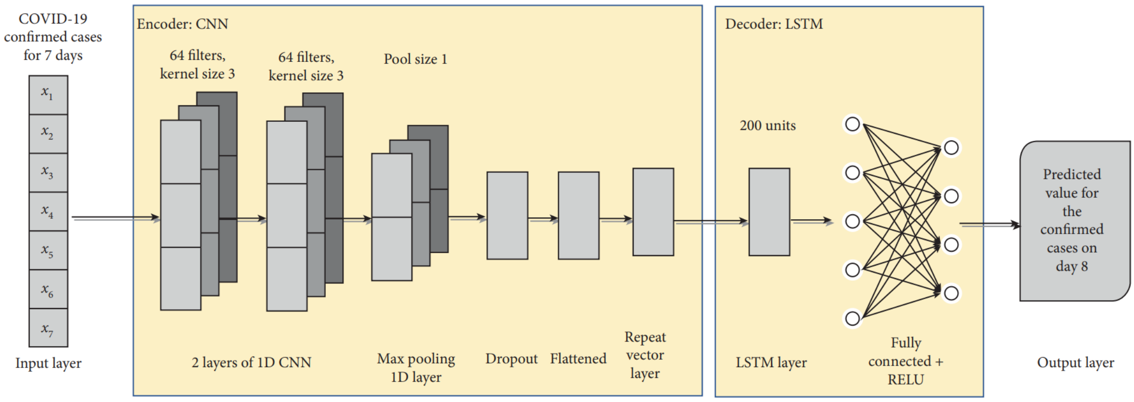


Figure 8: The architecture of the specific CNN-LSTM model from Zain et al. (2021).

It should be noted that the observed new-cases data $x = (x_1, \ldots, x_n)$ was min-max-normalized, i.e., the transformation

$$\text{min-max}(x_i) = \frac{x_i - \min(x)}{\max(x) - \min(x)},$$

was applied to x. The inverse transformation was done to the final prediction, using the same min and max value. Some sort of normalization of the input data is commonplace in machine learning models, as it transforms all features to the same scale.

Adaptive Moment Estimation (ADAM), a modified version of gradient decent, was used for training the model. The MSE was used as loss function. The learning rate was not specified, hence the default learning rate for ADAM was chosen, i.e., 0.001 (TensorFlow, 2022c). The batch size of the model specifies the number of input-output pairs to train on before updating the weights, while the number of epochs is defined as the number of times the whole training data set is fed to the network. Zain et. al used a batch size of 22 with 472 epochs. The choice was based on a grid search algorithm using the Optuna framework. The same number of epochs was initially adopted to the model in this thesis, and the (unspecified) dropout rate was set to 0.2 from initial testing.

The model implemented in this thesis had 347109 weights (or parameters) in total. Hence, it is quite a complicated function. On the other hand, at the end of the day, it is simply a function that takes in seven values, and outputs a single value. Mathematically, the general modeling task can be described in the following way: Considering each eight-day sequence $(x_i, \ldots, x_{i+7})$ of observations, find a function $\mathcal{F}(x_i, \ldots, x_{i+6} \mid P) = \hat{X}_{i+7}$ that minimizes the loss function

$$l(\mathcal{F}(x_i, \ldots, x_{i+6} \mid P) - x_{i+7}),$$

w.r.t. the large parameter space P, for all $i = 1, \ldots, n-7$. Both the Covid-19 data sets have clear weekly patterns, with higher prevalence in the weekdays than the weekends. The two SARIMA models were constructed with this in mind. Can a model with such a simple problem description learn this pattern as well? This is explored in Section 4.

Remark: Fitting a CNN-LSTM model will not guarantee the same parameter estimates each time since the fitting procedure involves a lot of randomness: Firstly, the parameters are randomly initialized before the training begins. Secondly, the dropout layer randomly excludes 20% of the data in each batch in each epoch. However, for an already fitted model, the prediction is deterministic, as the dropout rate is excluded when performing this operation (TensorFlow, 2022a).

Remark: Each of these stochastic parts has their own seed. To ensure reproducible, all these seeds must be fixed. The first function in the provided Python script, called `set_all_seeds`, fixates all the necessary seed values, and is applied at the beginning of each predictive function. Even after doing this, the results may vary depending on software and hardware. The exact specifications used to produce the results in this thesis are soon provided.

3.2 On the different ways of forecasting

While a general method for forecasting with the SARIMA models were established in Section 2.5, a forecasting method for the CNN-LSTM model has yet to be specified. In this thesis, two different methods were used to forecast with each of the models.

Taraldsen and Zain et. al. generated the forecast from their models in fundamentally different ways. Taraldsen used the SARIMA models to predict the next seven days directly, as in Section 2.5, while the CNN-LSTM model published by Zain et. al. predicted 28 days into the future. However, the exact method used to predict these 28 days was not specified. Judging by the accuracy of the model prediction, and the fact that the prediction seems to be as accurate for the final days of the prediction as for the first few days (see Figure 7 in the article by Zain et al. (2021)), the model has probably been given updated information of the actual test data along the way. An alternative is to update the model with the newest prediction, which is more similar to how the SARIMA models generate their h-step prediction directly. This section specifies these two prediction schemes.

3.2.1 Prediction scheme 1

A trained CNN-LSTM model takes seven consecutive amounts of new cases from the Covid-19 data set and returns the predicted amount on the eight, ensuing day. To predict the new cases on day nine, the amounts for the seven preceding days are needed as input. When it is specified that the CNN-LSTM model uses prediction scheme 1, the most recent input of the model comes from the test data, i.e., the actual value of the day that was previously predicted. In Figure 9, only the grey-colored squares are used as input in for prediction, and the red-colored $x$es where originally part of the test data. Hence, they were therefore and was therefore not seen by the model under training. Thus, the model gets updated information before generating the next prediction. In this fashion, a series of one-step predictions can be used to generate a h-step prediction. Note that h is limited to the amount of test data.

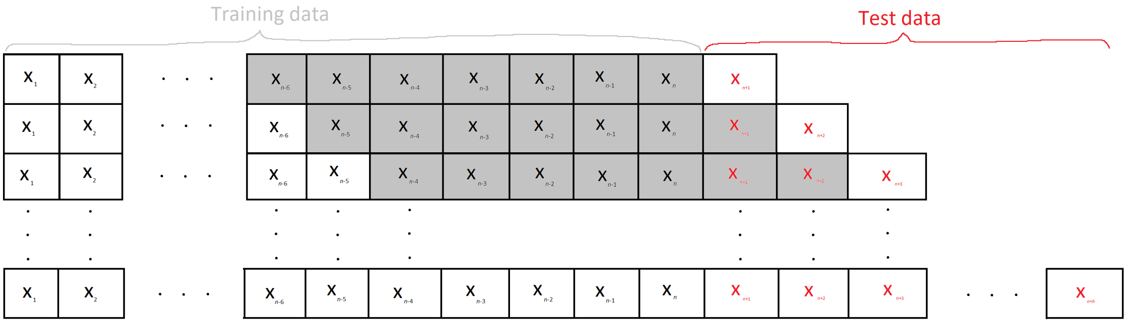


Figure 9: An illustration of prediction scheme 1 for the CNN-LSTM model.

The SARIMA models are much less computer intensive than the CNN-LSTM model. For this reason, it was decided that when the SARIMA models used prediction scheme 1, they make a series of one-step predictions, and fit a new model with all the available data at each time step. This is reflected in Figure 10, where the only difference from Figure 9 is that all data points preceding the next value to be forecasted are highlighted in grey.

Figure 10: Illustration of prediction scheme 1 for the SARIMA models.

3.2.2 Prediction scheme 2

Using prediction scheme 1 implied that the trained models preformed a series of one-step predictions with up-to-date information. When it is specified that prediction scheme 2 is used, the models are never exposed to the test data. For the CNN-LSTM model, a series of one-step predictions are executed to perform a h-step prediction. However, the predictions themselves are used as input to predict the next day, as opposed to the test data. This is illustrated in Figure 11, since the red test data is replaced by green $\hat{X}_i$s. When prediction scheme 2 is used for the SARIMA models, they simply perform the h-step prediction described in Section 2.5. This is similar to the CNN-LSTM model's prediction scheme 2 since the predictions replace the observations from the training data as the prediction unfolds (see Equation (12)). Eventually, the input of the forecasts using prediction scheme 2 are solely predicted values. By construction, this scheme should result in less precise predictions, and the prediction error should rapidly increase, as explained by Equation (13) for the SARIMA models. This is a more realistic prediction scheme for the task at hand, as one often wants to predict multiple steps ahead, to have time to form a response based on it.

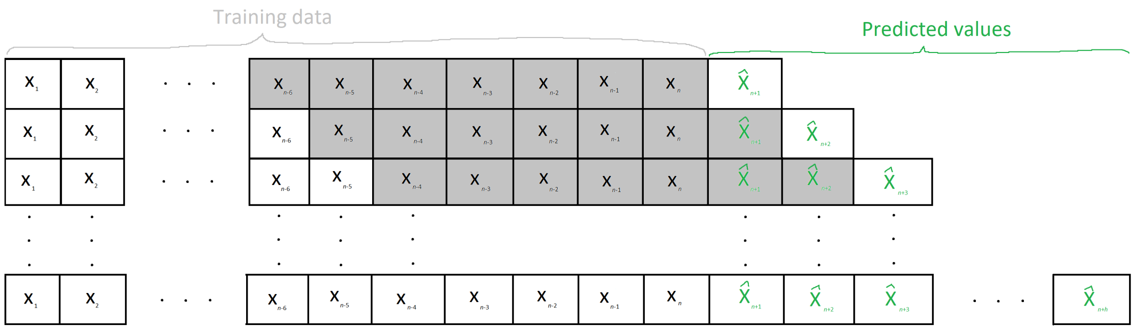


Figure 11: Illustration of prediction scheme 2 for the CNN-LSTM model.

3.3 Programming languages

Matlab was used to implement the SARIMA models and to create all the plots in the thesis. Python was used for the CNN-LSTM model, along with TensorFlow. Finally, R was utilized to pre-process the two data sets. The relevant hardware and software specifications are given in Table 1. All code used to produce the result of this thesis can be found in Appendix B.

Table 1: Relevant computer specifications.

Processor	Intel Core i7-3820
CPU	3.60 GHz
Operating system	Windows 10 Pro
Matlab version	R2022a
Python version	3.7
TensorFlow version	2.8
R version	3.6.1

4 Applying the models on Covid-19 data

In this section, the models were applied to the two Covid-19 data sets. Initially they were applied on the training and test sets used in the two introductory articles published by Zain et. al. and Taraldsen, to hopefully replicate their results. Concrete definitions of these data sets will now be given. **Taraldsen's data set** denotes the training set Taraldsen used to present the two SARIMA models. It consists of new cases in Norway from February 21st 2020 to November 10th 2020, which amounts to 264 days. The CNN-LSTM model was originally evaluated on **Zain's data set**, i.e., the global new cases from January 4th 2020 to July 17th, which is 196 days of data. The models were later compared on other partitions of both the Norwegian and the global data set. Finally, a simulation study was conducted to explore the behaviour of the models in more depth.

4.1 Data

The two data sets used in this thesis were introduced in Section 1. Some additional details about the data gathering process are now given.

The worldwide new cases data set, provided by WHO (2022), contains daily new Covid-19 cases from over 200 countries. A provided R-script was constructed to sum up the new cases on each day for each country, and can be found in Appendix B. This data set is called the global data set. The last day included in both data sets was February 20th 2022. The Norwegian data set started on February 21st 2020, while the global data set begun some weeks earlier, on January 4th. This amounts to 731 and 779 days of data for the two models, respectively. However, the last week of the data sets are especially uncertain and was ignored. Zain et. al. reference the same data set from WHO. According to their article, their data set contained new cases from 216 countries. However, at the time of writing, WHO's new cases data set contains data from 236 countries. This may cause minor deviations in the results compared to the results reported by Zain et. al.

4.2 Exploring the fit of the SARIMA models on the Taraldsen's data set

ARMA models assume stationary input data. To approach homoscedasticity, the input data for the SARIMA model and the Gandalf model was log-transformed, preceded by a $\max(\cdot, 0.1)$ operation to deal with zero valued inputs. The two difference operators $(1 - B)(1 - B^7)$ was applied to the log-transformed data. As remarked in Section 3.1, this transformed time series should be stationary to fulfil the SARIMA model assumptions. Figure 12 shows the result of applying these transformations to the Norwegian and the global data set. The first row shows the two time series. A rapid increase of the daily new cases can be observed in the last few months of both time series. For the log-transformed time series in the next row, the variance is drastically reduced, and the rapid increase in the final months of the data sets is not as notable. On the contrary, a fast upward trend can be observed the earlier parts of both data sets. Both time series in the last row of the figure look quite stationary, which implies that the difference operations did what they are supposed to. The exception is at some of the earlier dates, where both time series have a much larger fluctuations than for later dates. These fluctuations seem to be conditional on previous fluctuations since high fluctuations are followed by high fluctuations and vice versa.

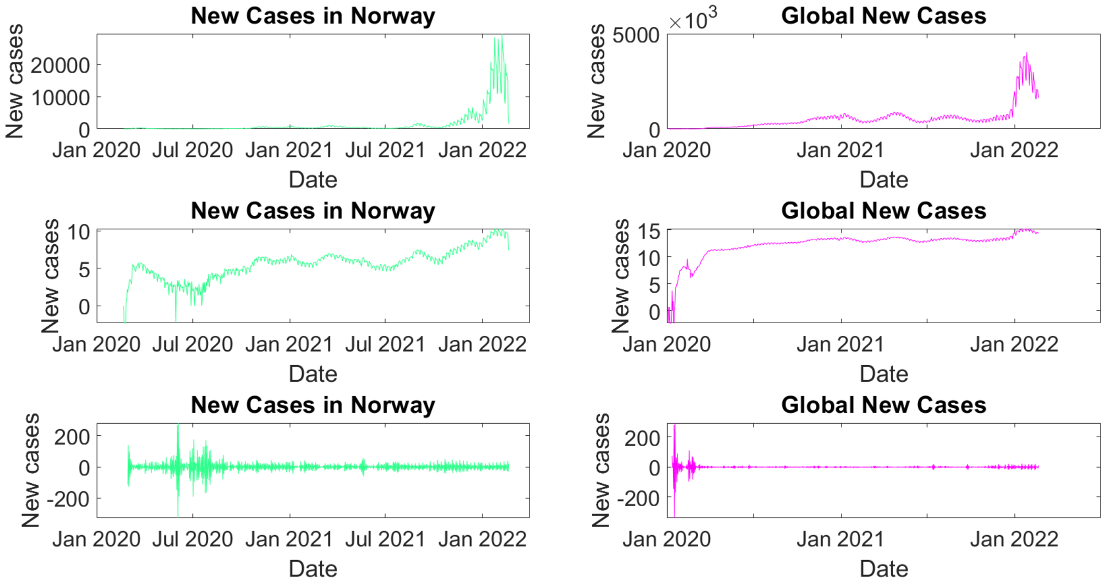
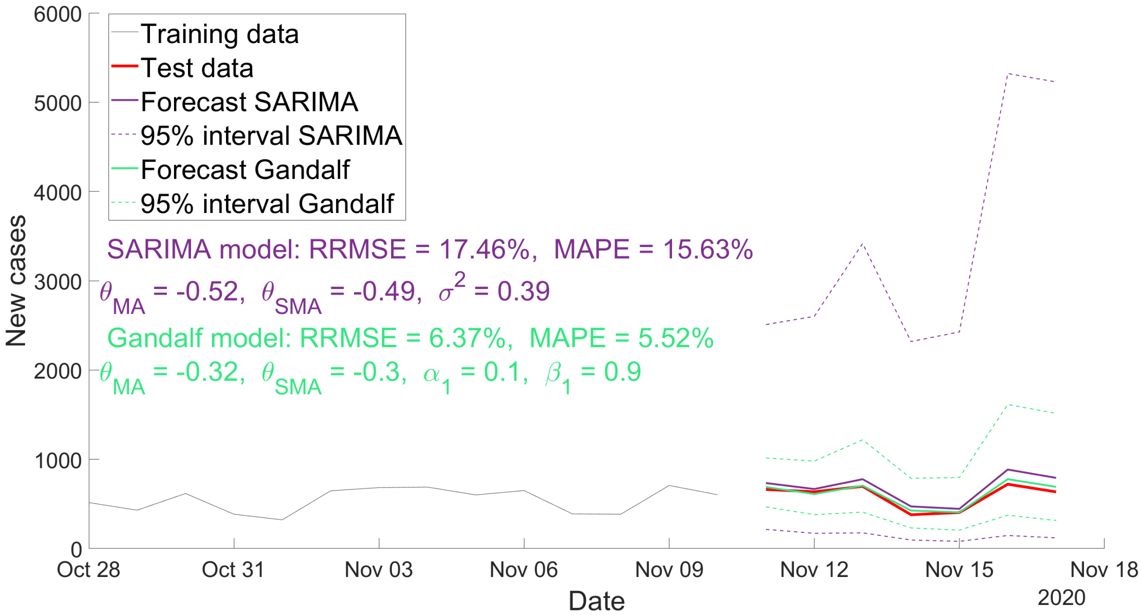
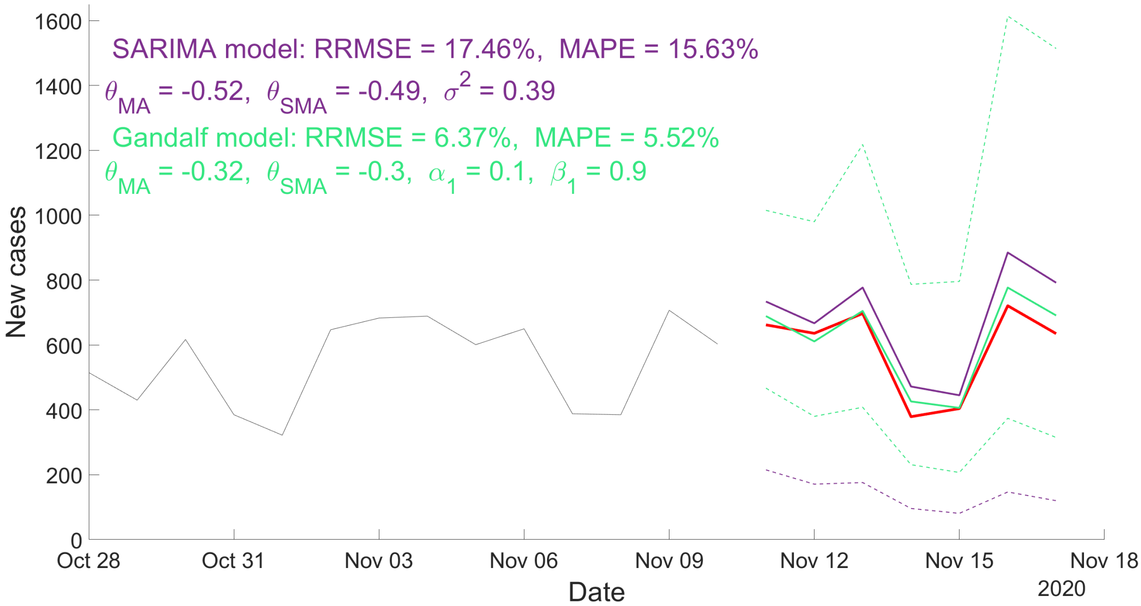


Figure 12: The Norwegian and the global new cases time series with transformations.

Figure 13 shows the seven-day forecasts from the SARIMA models when fitted on Taraldsen's data set. The plot includes the predictions and the prediction intervals for the SARIMA model and the Gandalf model. Like Taraldsen, prediction scheme 2 was used to generate the seven-day forecasts of new cases from November 11th to November 18th. The accuracy measures and the parameters estimates have been included in the plot. The forecast from the Gandalf model is slightly closer to the observed values than for the SARIMA model, as indicated by both measures of accuracy. The prediction interval corresponding to the Gandalf model is narrower than the SARIMA model's, almost by a factor of four. The parameter estimates for both models were also included and are close to Taraldsen's estimates. These minuscule differences can be explained by small adjustments in the data set provided by FHI since Taraldsen extracted the data set. Figure 2 in Taraldsen's article was used for reference.



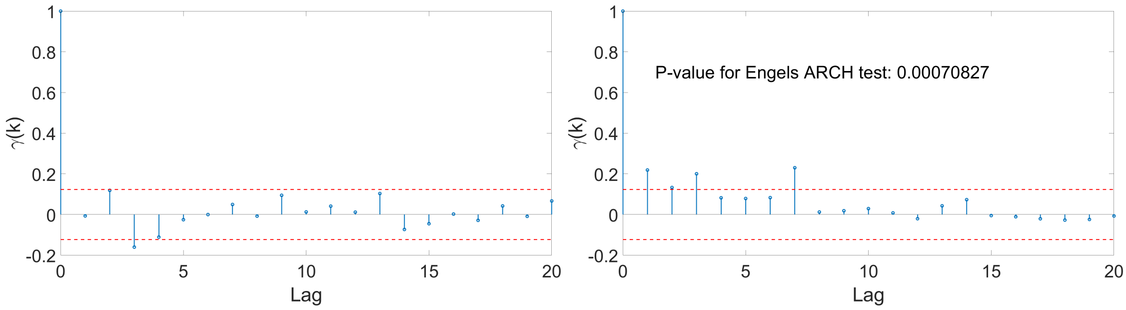
(a) seven-day forecast of Norwegian data from November 11 2020



(b) Zoomed version of (a). The upper limit of the prediction interval for the SARIMA model is not in frame

Figure 13: Replication of results from Taraldsen (2020b) with the SARIMA model and the Gandalf model.

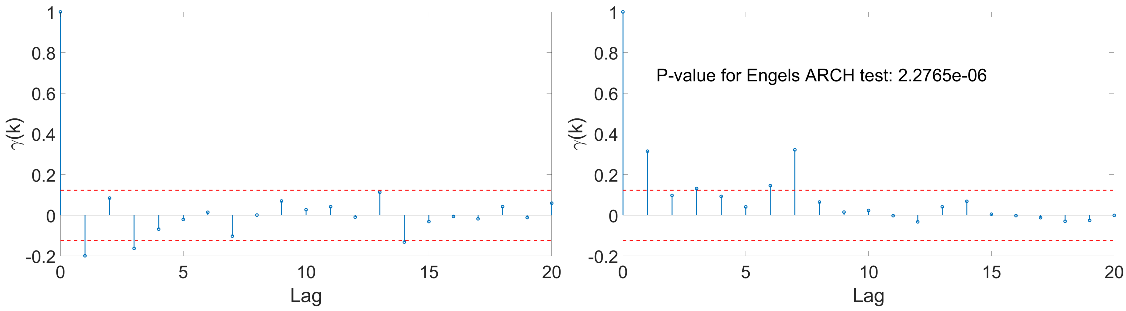
The better performance of the Gandalf model on the Taraldsen's data set suggests that the underlying noise during this partition of the Norwegian data set is closer to GARCH(1,1) noise than Gaussian noise. This was explored further by looking at the ACF plot of the residuals in Figure 14.



(a) Normal **(b)** Squared

Figure 14: ACF plot of residuals and squared residuals for SARIMA model on Taraldsen's data set.

The ACF plot of the residuals in Figure 14(a) barley shows one significant lag, indicating that the SARIMA model fits the data well. Some lack of fit can be observed in Figure 14(b). The p-value produced by Engel's ARCH test with two significant lags (to test for GARCH$(1, 1)$ noise) is lower than 0.05. These results suggest conditional heteroscedasticity in the training data, and motivates the application of the Gandalf model to the original Norwegian data set. The ACF plots of the squared residuals from the Gandalf model can be seen in Figure 15(a). The plot displays three significant lags, specifically at lag 1, 3 and 14. While they are barely significant, this is worse than for the SARIMA model. Additionally, the p-value generated by the ARCH test on the residuals is even lower than its SARIMA equivalent.



(a) Normal **(b)** Squared

Figure 15: ACF plot of residuals and squared residuals for Gandalf model on Taraldsen's data set.

4.3 Exploring the fit of the CNN-LSTM model on Zain's data set

In the article from Zain et al. (2021), the CNN-LSTM model was initially trained in Zain's data set. As remarked in Section 3.1.3, the process of fitting a CNN-LSTM model involves randomness. Thus, the predictions may vary each time the model was fitted. To account for this, Zain et. al fitted their model ten times. With each of the ten fitted models, a 28-day forecast with prediction scheme 1 was generated. From these forecasts, ten RRMSE scores and MAPE scores can be calculated. Zain et. al. reported the mean of these ten RRMSE scores and of the ten MAPE scores to be 5.30 and 0.19, respectively. By inspection of the prediction and the test data from Figure 7 in their article, the mean MAPE score seems way too low, and was not used for reference in this thesis.

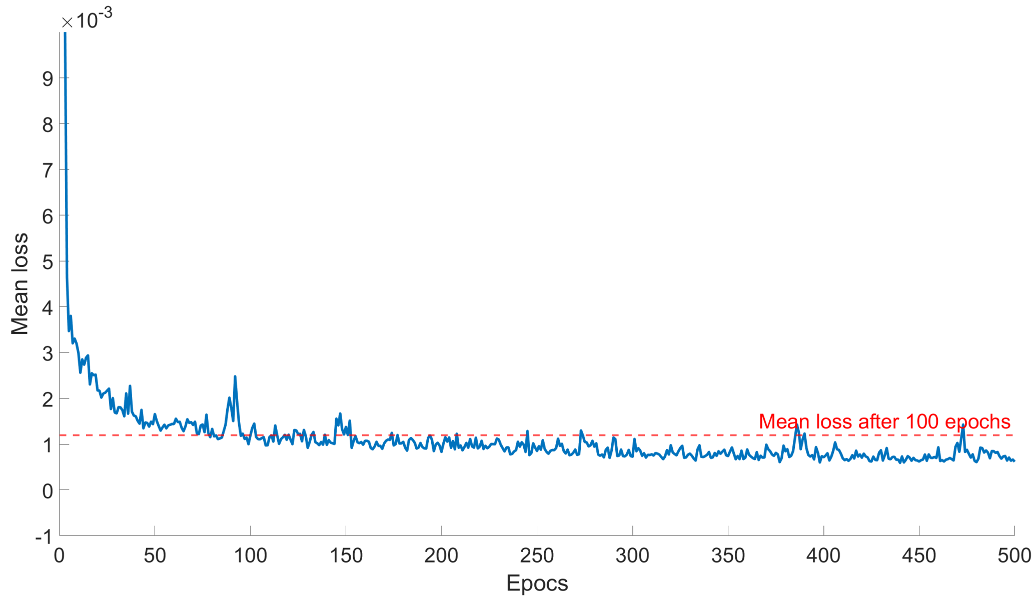


Figure 16: Mean loss function of ten equally specified CNN-LSTM models, trained on Zain's data set.

Initially, 472 epochs were run to fit the ten CNN-LSTM model. However, this resulted in a mean RRMSE of 8.35, which is worse than Zain et. al. The loss function for the models might explain why. Figure 16 shows the mean of the ten loss functions (one for each model). The red line is the value of the mean loss function at 100 epochs. Observe that the mean loss quickly decreases to around 0.0015. While it continues to decrease slightly afterwards, using all the 472 epochs might cause overfitting. To prevent this, the number of epochs for all CNN-LSTM models used in this thesis was reduced to 100. A more practical benefit of this change was that the time used to fit each model was substantially reduced. Figure 17 shows the predictions of the ten CNN-LSTM models as specified above, together with a RRMSE and MAPE score for each model. The mean RRMSE and MAPE from the ten models is also included. Observe than the mean RRMSE is lower than what Zain et. al reported.

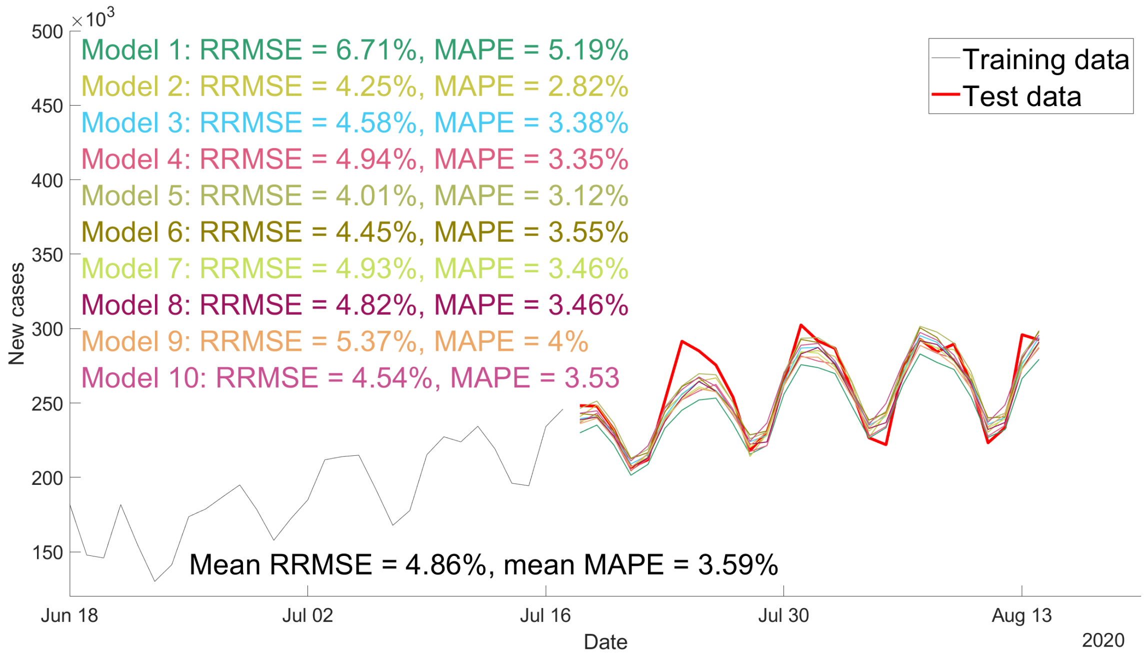


Figure 17: Forecast with ten CNN-LSTM models from July 18th to August 14th 2020 on the global new-cases data with prediction scheme 1. The models were trained on Zain's data set.

4.4 Comparing the three models

All three models performed well on the training and test sets used in their respective articles. This was to be expected; after all, they were constructed for those particular data sets. This section explores how good the models predict on their opposing original data sets, how they perform across other partitions of the data sets, and the effect on the models' performances with reduced samples sizes.

4.4.1 Comparisons on Taraldsen's data set

Initially, ten CNN-LSTM model were trained on Taraldsen's data set and used to forecast the new cases on the seven ensuing days. The forecasts are shown in Figure 18. Observe that the ten forecasts of the new cases on November 18th are more spread out than the foretasted values on November 11th. This is expected when using prediction 2.

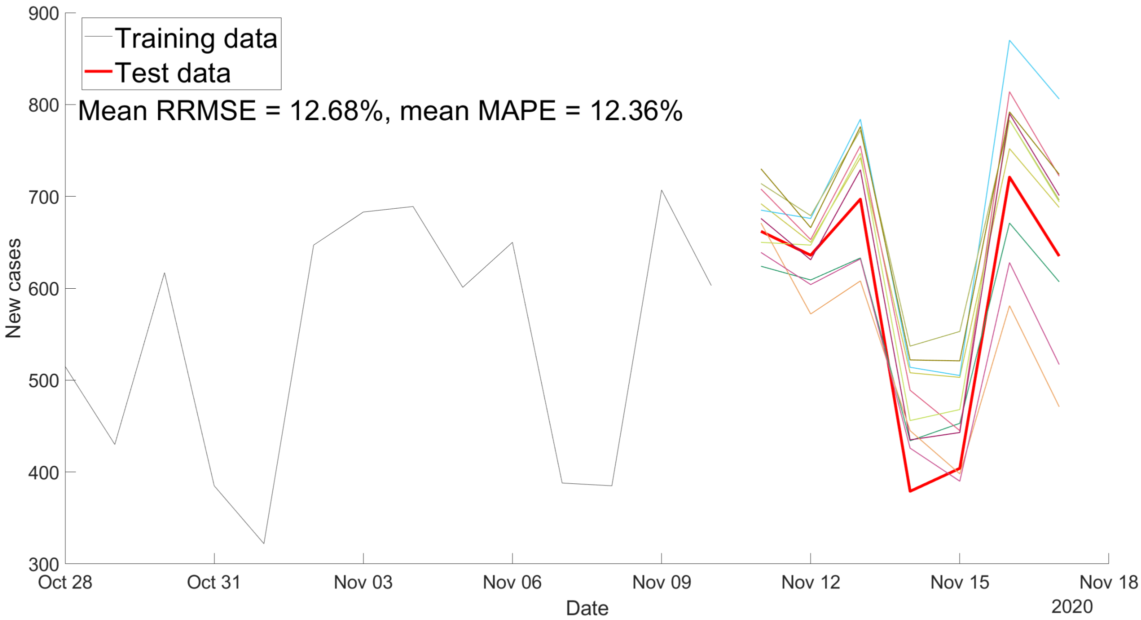


Figure 18: Forecast with ten CNN-LSTM models from November 11th to November 18th 2020 on the Norwegian new-cases data with prediction scheme 1. Taraldsen's data set was used for training.

To compare the SARIMA models with the CNN-LSTM models, it would have been possible to fit ten of each SARIMA models and generate one forecast for each of them. This would give ten forecasts for each model. However, since the fitting and prediction of the SARIMA models are deterministic operations (when a deterministic numerical optimization algorithm is used), this is not meaningful for these models. It is also impractical to have ten forecast as opposed to one. There are many possible ways to display one single prediction from the ten CNN-LSTM models. A tempting choice was to choose the CNN-LSTM model with the best RRMSE and MAPE. However, this can only be determined when the test data is known. Hence, a measure of central tendency of some sort was a better alternative. The mean was the measure of choice, calculated as the mean of each of the ten predicted values of each day. This will later be called **the mean CNN-LSTM prediction**. The accuracy of this single prediction must not be confused with the mean of the ten accuracy measures.

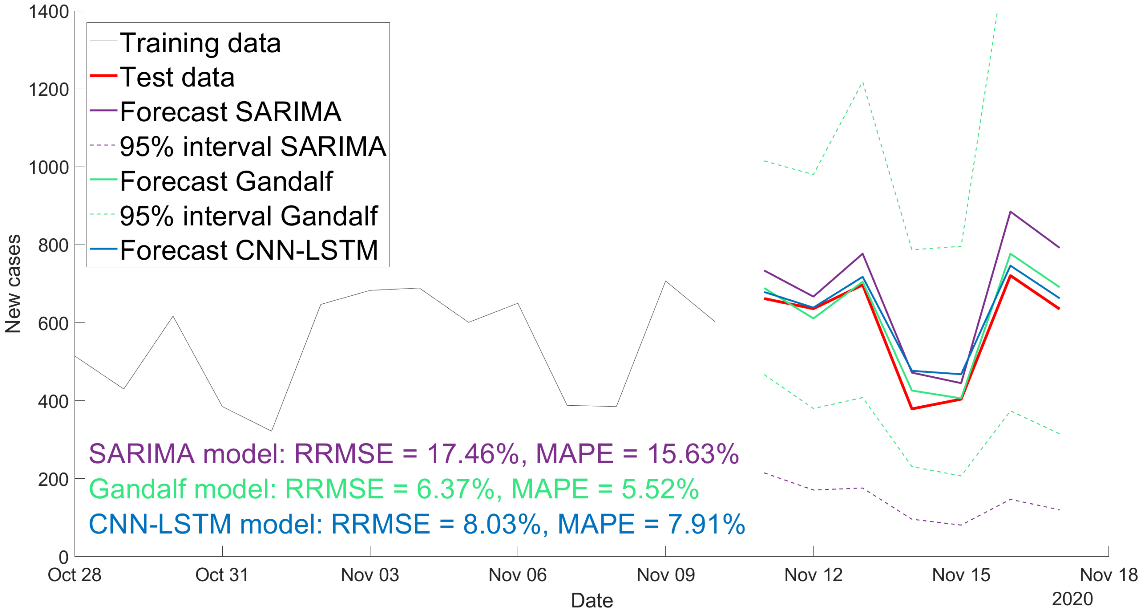


Figure 19: Forecast from the three models with prediction scheme 2, from November 11th to November 18th 2020 on the Norwegian new-cases data. Taraldsen's data set was used for training.

The three seven-day forecasts generated from the SARIMA model, the Gandalf model, and the mean CNN-LSTM prediction, can be seen in Figure 19. Note that the upper limit of the prediction intervals for the SARIMA model and the Gandalf model are mostly outside the plot. More importantly, note that there is no prediction interval for the machine learning model. The Gandalf model performed best with respect to the two accuracy measures, and the CNN-LSTM model was more accurate than the SARIMA model.

Zain et. al. presented the CNN-LSTM model with a 28-day prediction using prediction scheme 1. One such prediction was generated by all three models, after they were trained on Taraldsen's data set. These 28-day forecasts are shown in Figure 20. The Gandalf model gave the most accurate forecast, and the SARIMA model trailed closely behind. The CNN-LSTM model was worse than its competitors. The ten individual predictions from the CNN-LSTM model can be found in Appendix A (Figure 41).

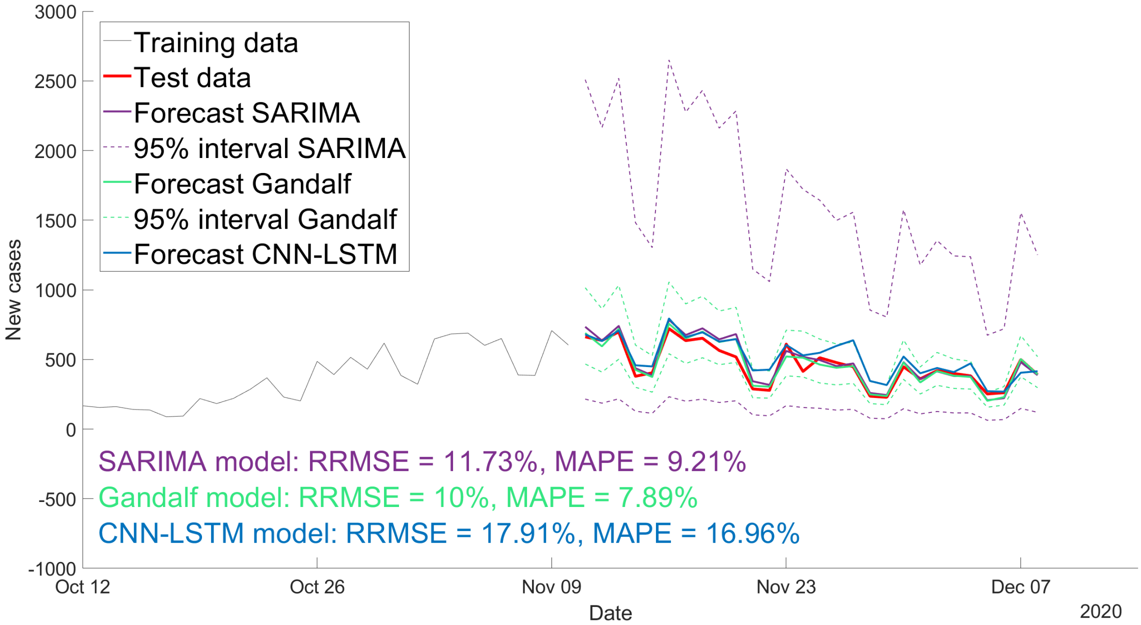


Figure 20: Forecast from the three models with prediction scheme 1, from November 11th to December 8th 2020 on the Norwegian new-cases data. Taraldsen's data set was used for training.

The time to fit and predict with each model may also be an important factor to consider when differentiating the models. To fit a single CNN-LSTM model to the above training data, Python used around 7.3 seconds, while generating a seven-day and 28-day forecast from a pretrained model took around 1.27 and 2.55 seconds, respectively. Each of these processes are repeated ten times. The SARIMA model took 1.3 seconds to fit and used 0.04 and 14.39 seconds to generate seven and 28-day forecasts on the fitted model, respectively. To fit the Gandalf model on Taraldsen's data set took around 1.30 seconds, while the forecasts took 0.05 and 37.90 seconds, respectively. The large deviation from the plain SARIMA model is a result of the slightly longer time to fit. When using prediction scheme 1, this difference increases with the length of the forecast, since they fit a new model on each consecutive day. The results are summarized in Table 2. Note that these time measurements depend on a lot of factors, like the amount of samples in the data and the hardware used. However, for the sample sizes used in this thesis, the relative differences in time between the models did not change notably.

Table 2: Time used to fit and forecast with each of the models, trained on Taraldsen's data set.

Model	Time to fit	Seven-day forecast	28-day forecast
SARIMA	0.32 seconds	0.04 seconds	14.39 seconds
Gandalf	1.30 seconds	0.05 seconds	37.90 seconds
CNN-LSTM	64.57 seconds	12.67 seconds	25.45 seconds

When using prediction scheme 1 to predict the 28 days ensuing Taraldsen's data set (i.e., November 11th to December 8th 2020), the SARIMA models estimate a new set of parameters each time a new value from the test data is available. Figure 21 displays the change in parameter estimates for each of the parameters of the Gandalf model. No significant deviation from the estimates obtained on the original training set could be observed in any of the four parameters. This is not that surprising, since adding 28 days to the original 264 days is a relatively small addition. A possible linear upward trend can be observed for the estimated α_1, mirrored by a linear decrease in the β_1 estimates. These may eventually deviate significantly from their original estimates.

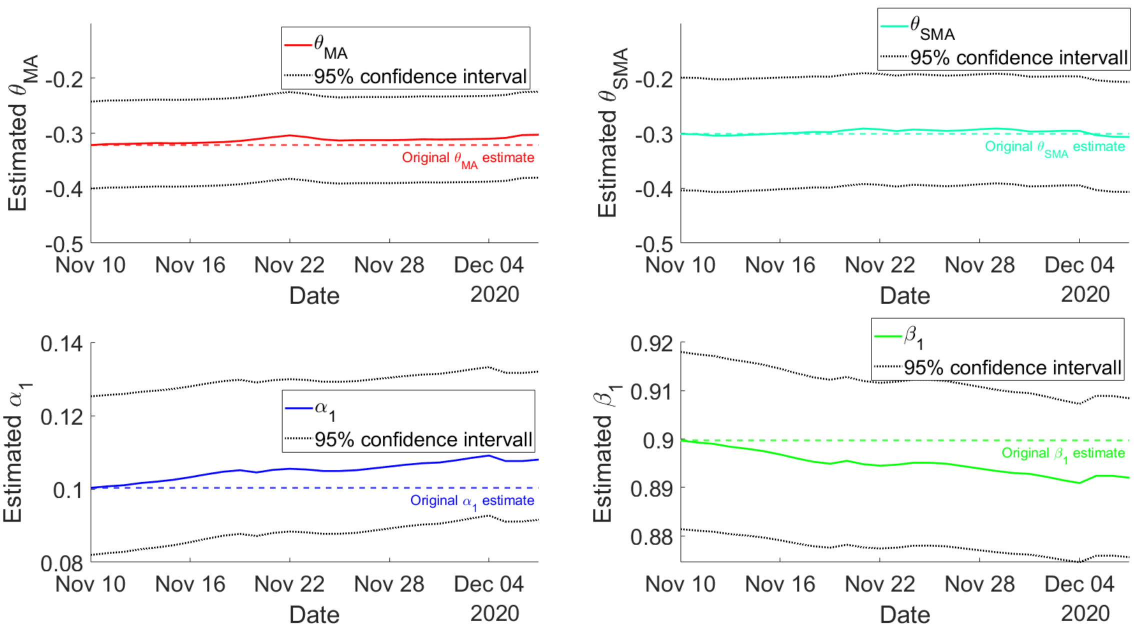


Figure 21: The parameter estimates from the Gandalf models used to predict the 28 days shown in Figure 20. The dates along the x-axis is the last day used to train the model used to predict the day after. The colored dotted lines are the parameter estimates of the Gandalf model fitted on Taraldsen's data set.

Figure 22 displays the equivalent plot for the SARIMA model. While the estimates for θ_{MA} and θ_{SMA} are stable, the σ^2 estimates follows a clear linear trend, eventually reaching a significant difference from the original σ^2 estimate.

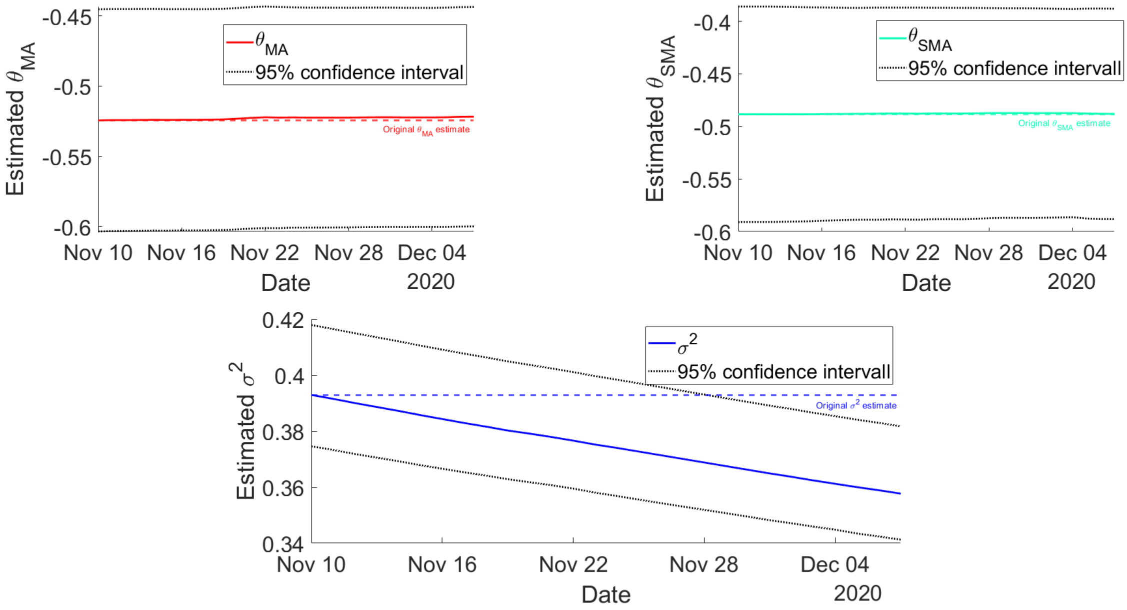


Figure 22: The parameter estimates from the SARIMA models used to predict the 28 days shown in Figure 20. The dates along the x-axis is the last day used to train the model used to predict the day after. The colored dotted lines are the parameter estimates of the SARIMA model fitted on Taraldsen's data set.

4.4.2 Comparisons on Zain's data set

The two SARIMA models were tested on Zain's data set. Figure 23 shows the seven-day forecasts from the SARIMA models, while Figure 24 shows the ACF plots of their residuals and their squared residuals. Both models were more accurate than they were on Taraldsen's data set, and the SARIMA model scored slightly lower than the Gandalf model on both measures. Some of the parameter estimates are quite different from Taraldsen's data set. In particular, the estimated variance for the SARIMA model were almost twice as high, and the estimated θ_{MA} and θ_{SMA} are quite different between the Gandalf models. The SARIMA model is more accurate than the Gandalf model, but it also has a much wider prediction interval.

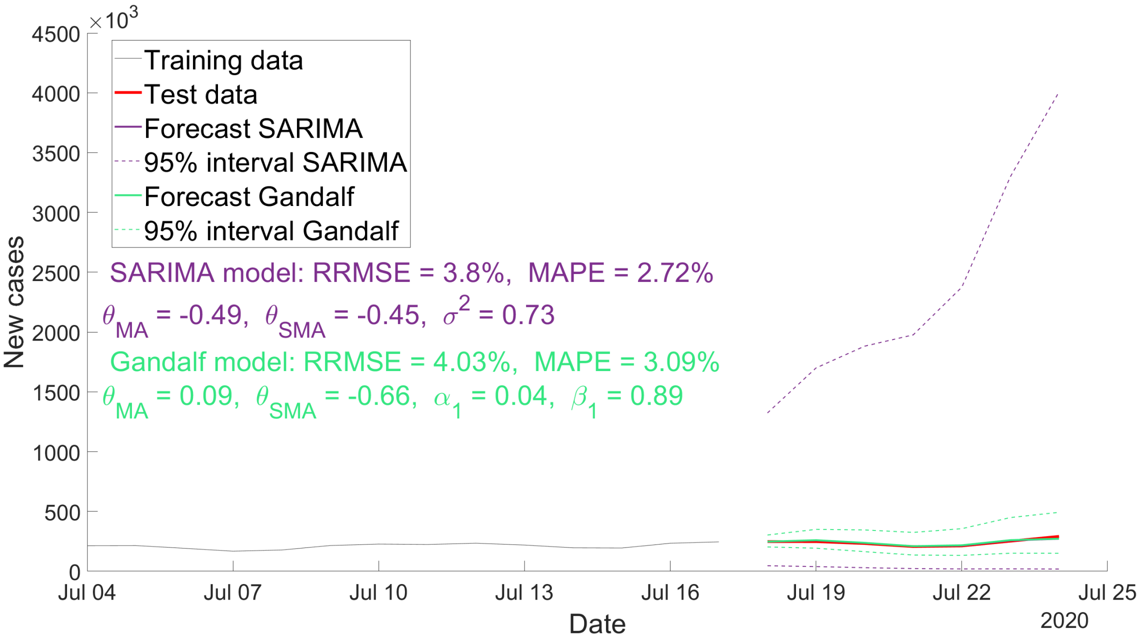
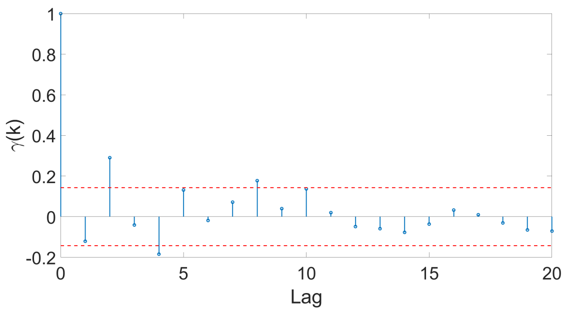
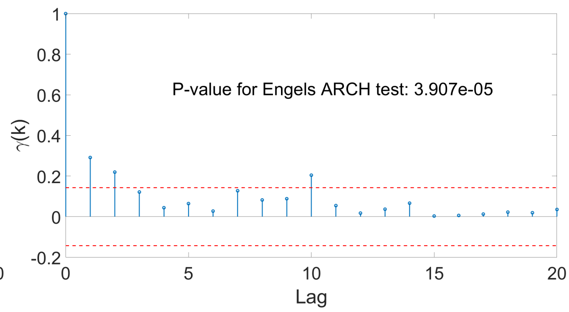


Figure 23: Forecast from the SARIMA models with prediction scheme 2 from July 18th to July 24th on the global data set. Both models were trained on Zain's data set.

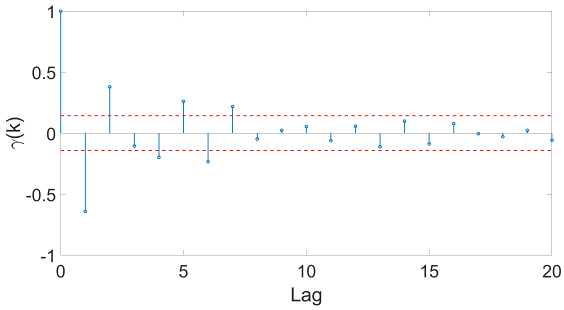
The ACF plot in Figure 24(a) of the residuals from the SARIMA model showed several significant lags, suggesting that some of the correlation in the data have not been captured by the model. The ACF of the Gandalf-residuals (Figure 24(c)) were similar. When the residuals contain first order dependencies, it is not surprising that the squared residuals are serially dependent (see Figure 24(b) and Figure 24(d). The added GARCH$(1, 1)$ model for the noise component was not enough to account for this. Engel's ARCH test supports these inferences, with p-values close to 0 for both the SARIMA and the Gandalf model's residuals.



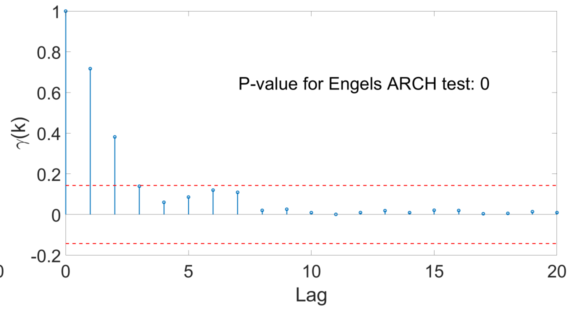
(a) ACF of residuals from the SARIMA model



(b) ACF of squared residuals from the SARIMA model



(c) ACF of residuals from the Gandalf model



(d) ACF of squared residuals from the Gandalf model

Figure 24: ACF plot of residuals and squared residuals for the SARIMA model and the Gandalf model, trained on Zain's data set.

Figure 25 and Figure 26 shows the seven-day and the 28-day forecasts from the three models trained on Zain's data set, respectively. The seven-day forecast used prediction scheme 2 (like Taraldsen), while the 28-day prediction used prediction scheme 1 (like Zain et. al). On the seven-day forecast, both SARIMA models obtain a slightly lower RRMSE score than the machine learning model. However, all results are excellent according to Zain et al. (2021) and Li et al. (2013). Note that the prediction interval for the SARIMA model is outside the field of interest, but can be observed in Figure 23.

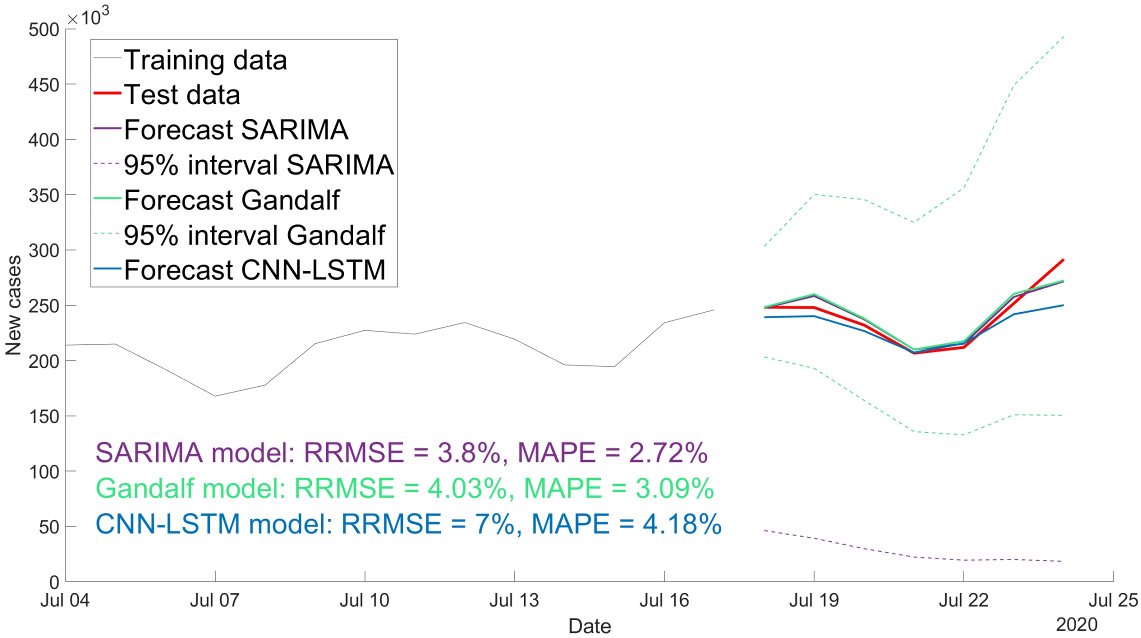


Figure 25: Forecast from the three models with scheme 2, from July 18th to July 25th 2020 on the global new-cases data. All models were trained on Zain's data set.

On the 28-day forecasts, all three models obtain a slightly lower RRMSE score than what Zain et. al. reported with their CNN-LSTM model. Note that the RRMSE and the MAPE for the mean prediction are slightly lower than the mean of the RRMSE and MAPE for the ten models in Figure 17.

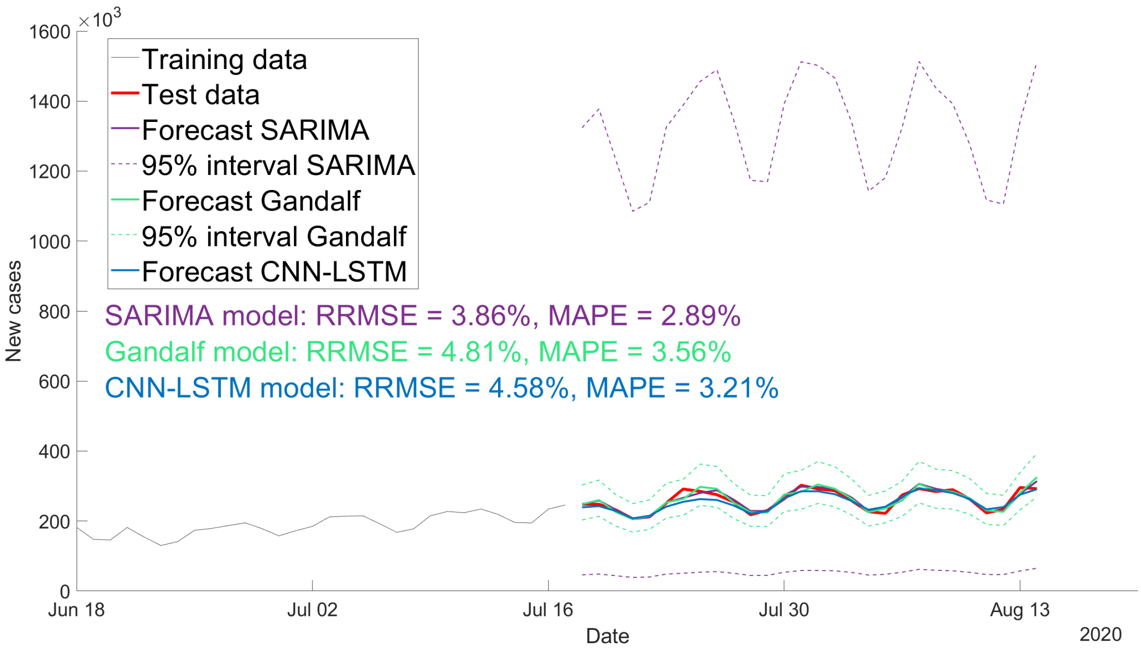
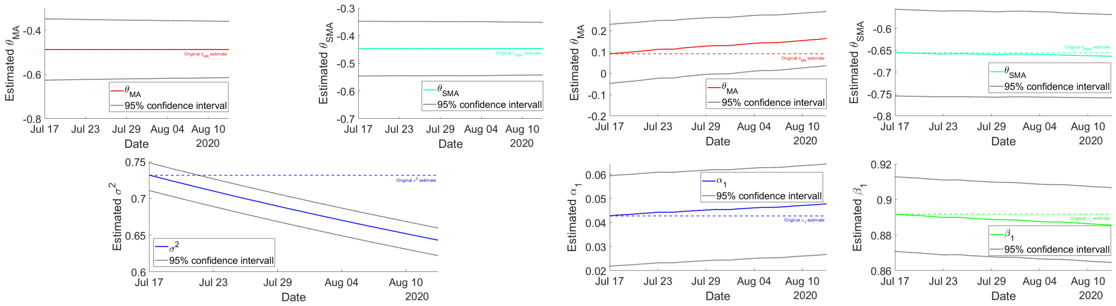


Figure 26: Forecast from the three models with scheme 1, from July 18th to August 14th 2020 on the global new-cases data. All models were trained on Zain's data set.

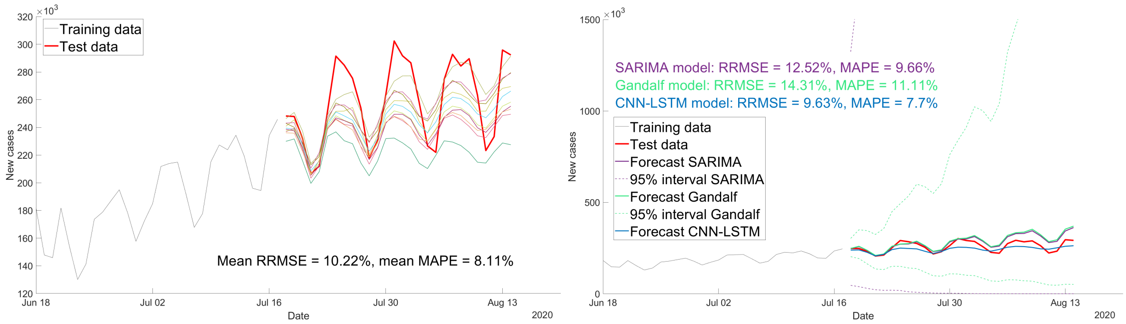
The 28 sets of estimated parameters for both models were collected and plotted in Figure 27. Similar phenomena to their equivalent plots based on Taraldsen's data set can be observed. However, the deviation of the σ^2 estimate is even more drastic.



(a) The parameter estimates from the SARIMA model. **(b)** The parameter estimates from the Gandalf model.

Figure 27: The parameter estimates for the two SARIMA models used to predict the 28 days shown in Figure 26. The dates along the x-axis is the last day used to train the model used to predict the day after. The colored dotted lines are the parameter estimates of the SARIMA model fitted on Zain's data set.

As a curiosity, 28-day forecasts with prediction scheme 2 were compared between the three models. The prediction from the ten individual CNN-LSTM models, in Figure 28(a), seems to converge to straight lines. This suggests that the CNN-LSTM model predictions does not maintain the weekly structure of the test data for longer forecasts. For the mean CNN-LSTM prediction in Figure 28(b), the same phenomenon is amplified. On the contrary, the forecasts from the two SARIMA models keep the weekly structure throughout their forecasts. Surprisingly, the machine learning model was more accurate than the SARIMA models. Naturally, the prediction intervals for the SARIMA models expand since prediction scheme 2 was used.

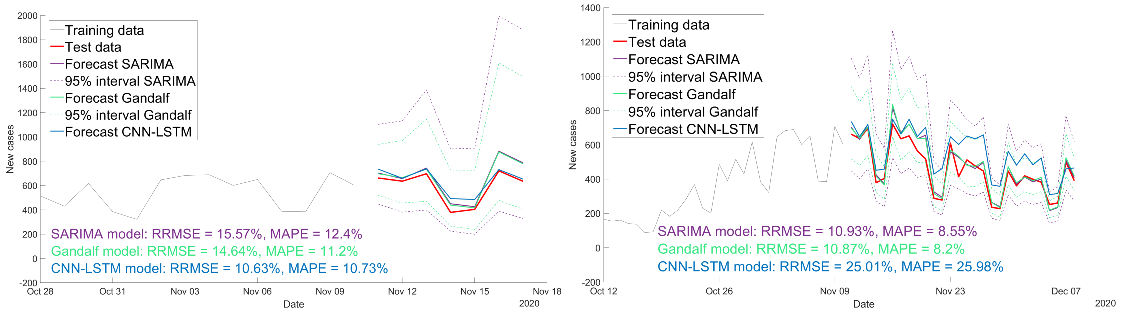


(a) **(b)**

Figure 28: Forecast from the three models with scheme 2, from July 17th 2020 on the global data set. The models were trained on Zain's data set.

4.4.3 Comparisons with less training data and at different partitions

Until now, all data available before the final day of the training data sets have been used under training. More data is usually preferred in statistics. On the other hand, the handling of a new positive Covid-19 test has changed throughout the pandemic. Say, for example, that in an earlier period, a newly infected individual and all people close to them were obligated to stay at home, whereas in a later period, a new ruling says that neither the infected nor their closest people need to stay home if they did not feel ill. Giving the models information from both of these periods might confuse then, and hence, make them more inaccurate.

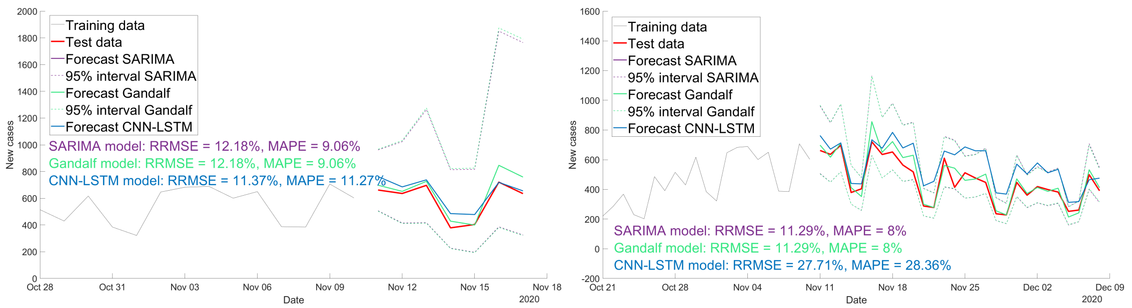


(a) Seven-day forecast

(b) 28-day forecast

Figure 29: Forecasts using the 100 previous Norwegian data training data from November 10th 2020.

Figure 29 displays the seven-day and 28-day forecasts from the three models when reducing the training data set to the 100 days prior to November 10th on the Norwegian data set. The most notable differences are that the Gandalf model is less accurate on the seven-day forecast, and that the machine learning model is less accurate on the 28-day forecast. The CNN-LSTM model is more accurate than the SARIMA models on the seven-day forecast, but less accurate on the ensuing 28 days. Both the forecasts and the associated prediction intervals from the SARIMA models are closer together than in Figure 19 and Figure 20, where all 264 data points were utilized. Overall, the results are worse than with all 264 data points, but not by much.

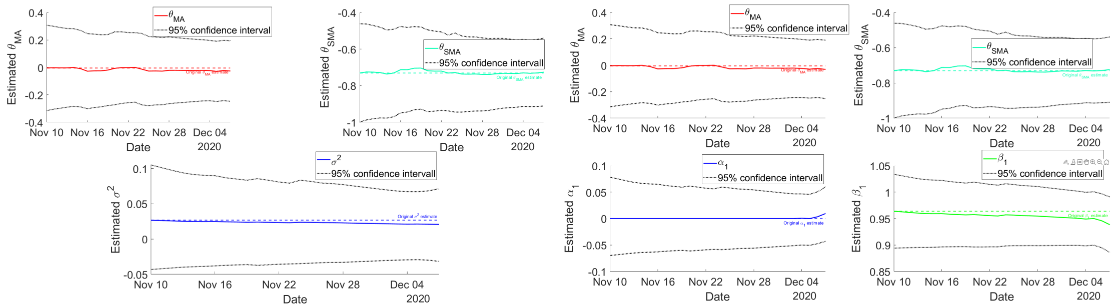


(a) Seven-day forecast

(b) 28-day forecast

Figure 30: Forecasts using the 50 previous Norwegian data training data from November 10th 2020.

Figure 30 displays the same forecasts, but trained on the 50 data points prior to November 10th 2020. The measures of accuracy are all higher than in Figure 19 and Figure 20, were all previous data was used. The accuracies are close to what is displayed in Figure 29. The SARIMA models are inseparable, as both their predictions and their prediction intervals align perfectly. The associated parameter estimates across the 28-day forecasts can be seen in Figure 31. As opposed to the earlier plots from Taraldsen's data set, the estimated σ^2 is stable, and the estimated θ_{MA} and θ_{SMA} between the two models are almost identical. Also note that the estimated α_1 is close to zero, while β_1 stays close to 1.



(a) The parameter estimates from the SARIMA model.

(b) The parameter estimates from the Gandalf model.

Figure 31: The parameter estimates for the two SARIMA models used to predict the 28 days shown in Figure 30. The dates along the x-axis is the last day used to train the model used to predict the day after. The colored dotted lines are the parameter estimates of the SARIMA model fitted on Zain's data set.

The models have only been fitted on one particular ending date for each data set, while some experimentation has been done with the sample size. In the remaining paragraphs of this section, they will also be compared on the ending dates corresponding to day 100, 300, 500 and the last possible day, for both data sets. For all these ending dates, the models were fitted with all preceding data points, but also with the preceding 100 and 50 samples. In total, this gives 14 training sets from the Norwegian and 14 training sets the global data set. On each training set, the three models were evaluated on both a seven-day and a 28-day forecast. Note that the ending dates are different for the Norwegian data set than for the global data set, since the latter starts at an earlier date. Since the last week of the data set may have inaccurate measurements, the last possible day to include in test data was February 13th 2022 (for both data sets). Thus, the last possible day to train for a 28-day forecast is 28 days earlier than February 13th, i.e., January 16th 2022. For simplicity, this was also the last training day for seven-day forecasts. This amounts to a total of 56 forecasts for each model. In Table 5 in Appendix A, the mean prediction of the ten CNN-LSTM models universally achieves lower RRMSE and MAPE scores than the mean RRMSE and MAPE of the ten prediction, which was reported by Zain et. al. For this reason, only the RRMSE and MAPE of the mean prediction were compared to the other models in the following table. All the results are shown into Table 3.

Table 3: All forecast results from the SARIMA model and the Gandalf model, and the mean CNN-LSTM prediction. Note that the RRMSE and MAPE from the CNN-LSTM model is based on the mean prediction of ten models. All 28-day predictions used prediction scheme 1, while prediction scheme 2 was used for all seven-day predictions.

Data set	Last training date	Sample size	Days ahead prediction	SARIMA model		Gandalf model		CNN-LSTM model	
				RRMSE	MAPE	RRMSE	MAPE	RRMSE	MAPE
Norway	30.5 2020	100	7	118.52	100.00	118.52	100.00	64.23	88.08
			28	74.12	60.63	75.13	61.46	73.05	87.49
		50	7	87.02	56.39	87.02	56.39	64.52	40.73
			28	60.35	40.00	262.40	118.28	52.72	41.21
	10.11 2020	264	7	17.46	15.63	6.37	5.52	8.03	7.91
			28	11.73	9.21	10.00	7.89	17.91	16.96
		100	7	15.57	12.4	14.64	11.20	10.63	10.73
			28	10.93	8.55	10.87	8.20	25.01	25.98
		50	7	12.18	9.06	12.18	9.06	11.37	11.27
			28	11.29	8.00	11.29	8.00	27.71	28.36
	16.12 2020	300	7	17.22	13.71	16.82	13.25	14.18	11.83
			28	25.26	21.62	26.71	20.25	28.85	22.54
		100	7	19.03	15.16	19.89	16.02	7.41	6.65
			28	24.77	20.03	23.93	19.16	29.22	22.85
		50	7	16.24	12.70	16.24	12.70	18.28	14.00
			28	28.14	22.61	28.41	22.49	25.81	20.82
	04.07 2021	500	7	10.29	7.35	14.15	10.82	7.85	7.72
			28	14.48	11.23	14.67	12.02	17.06	11.72
		100	7	5.55	4.09	7.79	5.05	11.56	11.95
			28	14.65	11.62	14.53	10.95	30.38	20.10
		50	7	21.25	17.26	20.87	16.97	10.79	9.23
			28	15.88	12.51	15.88	12.32	35.02	23.32
	16.01 2022	696	7	8.66	6.84	8.39	6.83	23.05	21.81
			28	10.93	11.06	9.41	9.38	18.76	16.57
		100	7	9.43	7.72	11.18	8.98	24.04	18.24
			28	8.90	8.78	9.19	9.12	29.56	30.89
		50	7	12.97	11.03	3.59	3.08	23.76	22.69
			28	8.86	8.51	8.87	8.28	45.55	46.06
Global	12.04 2020	100	7	30.39	28.25	35.32	31.34	10.72	9.85
			28	6.76	5.52	6.95	5.45	7.34	6.33
		50	7	15.65	14.44	15.63	14.42	14.38	13.47
			28	6.24	5.19	6.33	5.32	9.86	8.51
	17.07 2020	196	7	6.77	5.67	6.06	4.86	8.01	6.48
			28	3.86	2.89	4.81	3.56	4.58	3.21
		100	7	4.10	3.31	4.11	3.22	5.90	3.59
			28	4.07	3.14	4.06	3.06	4.17	3.20
		50	7	4.27	3.79	4.26	3.71	4.90	3.04
			28	4.29	3.45	4.09	3.21	3.97	3.12
	29.10 2020	300	7	6.79	6.16	4.75	3.52	11.19	9.90
			28	5.10	4.15	6.54	4.79	12.19	11.07
		100	7	6.79	6.16	6.61	6.01	7.97	6.79
			28	5.12	4.23	5.14	4.16	9.03	7.59
		50	7	6.69	5.88	6.67	5.85	9.44	8.41
			28	5.64	4.44	5.74	4.44	10.46	9.34
	17.05 2021	500	7	3.95	3.20	10.15	9.26	13.36	12.99
			28	4.36	3.61	5.82	5.19	9.29	7.89
		100	7	6.44	5.53	7.08	6.18	20.43	20.33
			28	4.61	3.92	4.71	4.00	15.30	13.83
		50	7	2.84	2.34	2.86	2.37	18.24	17.88
			28	4.56	3.84	4.58	3.86	38.96	39.08
	14.01 2022	725	7	14.81	11.74	10.49	9.42	12.17	12.08
			28	8.56	6.98	7.28	5.96	20.59	20.08
		100	7	12.24	10.52	13.64	10.86	10.35	10.30
			28	7.06	6.02	7.30	5.93	20.64	20.11
		50	7	12.47	10.66	13.16	10.39	11.38	10.64
			28	6.62	5.80	6.64	5.49	85.18	82.83

To get a grasp of the results in the above table, all the MAPE scores for each model were plotted in Figure 32. Some outliers can be observed for all models, but they mostly score below 25. The MAPE scores for the machine learning model seem to be a bit more spread out compared to the SARIMA models.

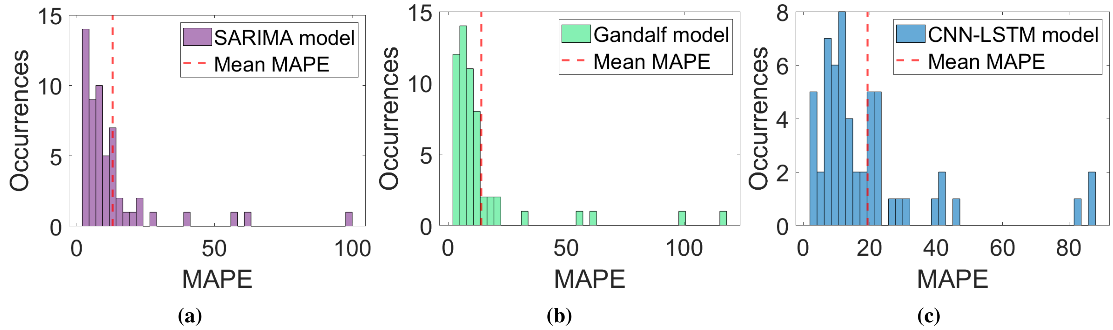


Figure 32: MAPE scores for each model from Table 3.

The MAPE scores of the models were examined further in Figure 33. To account for the outliers shown in Figure 32, the median MAPE for different subgroups of the above table, rather than the mean. Figure 33(a) shows that the SARIMA model obtains the lowest median MAPE score, with the Gandalf model closely behind. In Figure 33(b), the models were compared on a seven-day and a 28-day forecast. The SARIMA model scores lower on the 28-day forecasts, while the Gandalf model gave the best performance on the seven-day forecasts. The machine learning model struggles substantially more on the 28-day forecasts. Figure 33(c) compared the MAPE between the Norwegian and the global data set. All models did significantly better on the global data set, but the SARIMA models obtained an almost twice as low MAPE score as the machine learning model on this data set. Figure 33(d) demonstrates that reducing the sample size to 100 did not affect the MAPE much. When only 50 data points were available, the SARIMA models received lower MAPE scores than previously, while the CNN-LSTM models increased slightly.

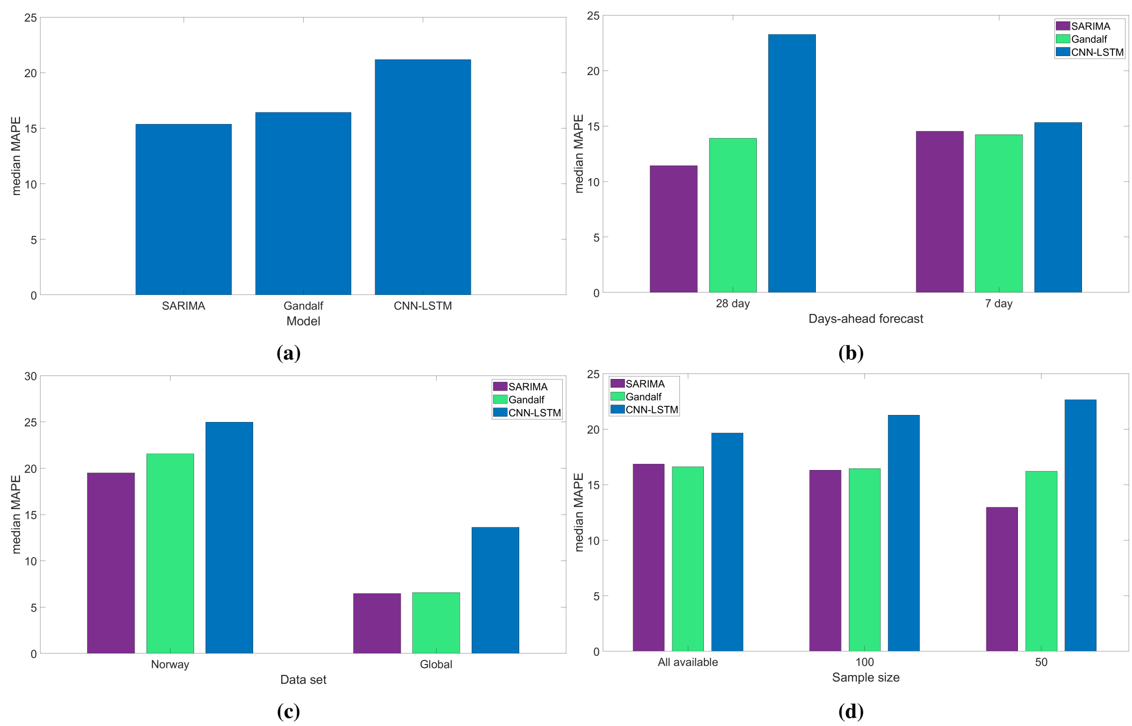


Figure 33: Comparisons of models based on MAPE scores from Table 3

The same analysis was also done on the RRMSE scores from Table 3. These results were placed in

Appendix A (Figure 42 and Figure 43), since they gave more or less the same information as the analysis of the MAPE. The main difference from the MAPE scores is that the Gandalf model performed worse relative to the other two models, and that the CNN-LSTM model had a slightly better performance than the other two on the seven-day forecasts.

4.5 The models on simulated realizations

While no analytical expression for constructing a probability interval of the CNN-LSTM model exists, by using a parametric bootstrap technique, it is possible to estimate a distribution around the RRMSE scores and the MAPE scores for the models. The spread of this distribution is an alternative measure of the spread of the predictions from the CNN-LSTM model. Assuming that the time series are stochastic in nature and that the underlying model is known and can be simulated from, it would be possible to use the models to forecast on each of these simulations. This could be used to control the previous results. Maybe the SARIMA model was lucky in the prediction on the one known realization? Unfortunately, the underlying model for the Covid-19 new cases will forever be unknown. However, as the above analysis suggests, the Gandalf model was a good model for the time series. Using a Gandalf model trained on Zain's data set, 1000 simulated realizations of the global data from July 18th to August 14th 2020 were generated. The specific parameter estimates of this Gandalf model are equivalent to the estimates in Figure 23, i.e., $\theta_{MA} = 0.09$, $\theta_{SMA} = -0.66$, $\alpha_1 = 0.04$, and $\beta_1 = 0.89$. In Figure 34, three of these 1000 realizations were plotted together with the forecast, the theoretical prediction interval for the Gandalf model, and the 95% quantile range based on the 1000 realizations. Observe that the three realizations are contained inside the 95% prediction interval, and that the quantile range approximates the theoretical prediction interval quite well.

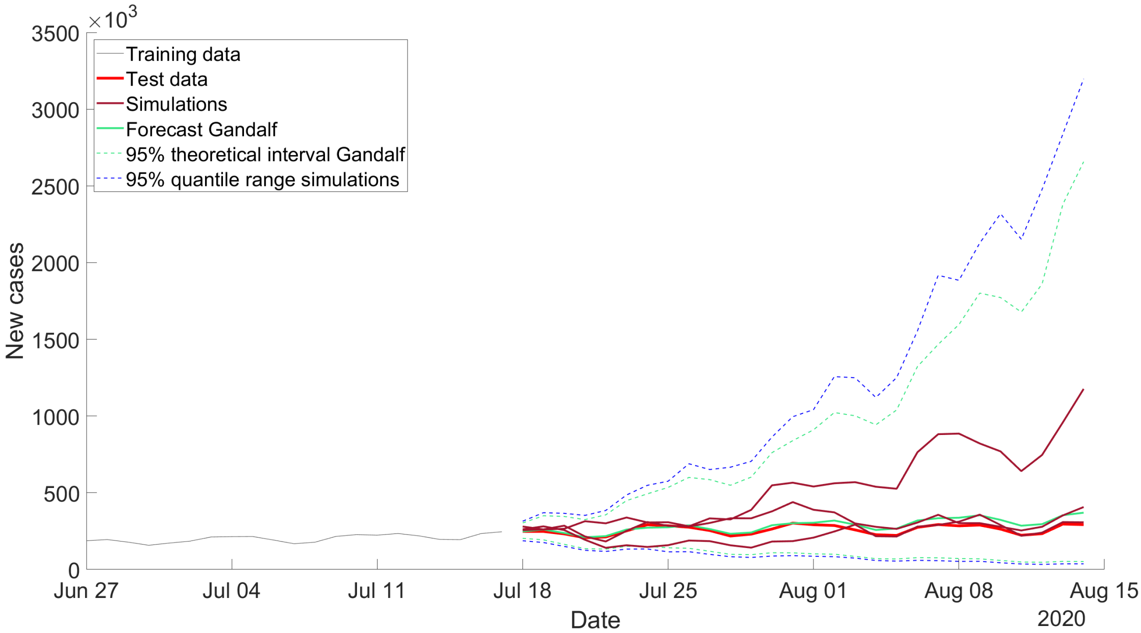


Figure 34: Three simulations from the Gandalf model. The model was trained on the original global training data. The 28 ensuing days were then simulated.

Under the assumption that the underlying process is the Gandalf model, each of these simulated realizations can be treated as a different realization of the global time series. Using prediction scheme 1, Figure 35 shows the prediction from the three models on the three realizations. Of particular interest was Figure 35(b), where the machine learning model struggled to keep up with the fast upward trend of the realization.

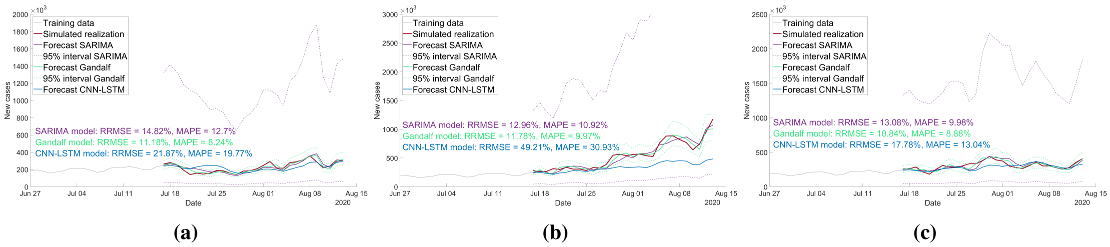


Figure 35: Predictions on three simulated realizations from the Gandalf model with prediction scheme 1.

For each of the 1000 simulations, one such forecast was made from each of the models, and the RRMSE and MAPE were evaluated. For 1000 simulated realizations, this gives 1000 RRMSE scores and 1000 MAPE scores for each model. Figure 40 shows the result of this operation for the Gandalf model, displayed as a histogram. The Gandalf model is expected to deliver the lowest RRMSE and MAPE scores, since it generated the "test data". Hence, these distributions serve as a baseline for the performance of the two other models. The measures of accuracy on the actual test data can also be seen in the plot. The Gandalf model was expected to deliver more accurate forecasts on the simulated realizations (from the Gandalf model) than on the test data. It would therefore be reasonable to expect the density of this histogram to center below the original accuracy measures. However, this is not the case. In fact, none of the scores were lower than the scores on the original test set.

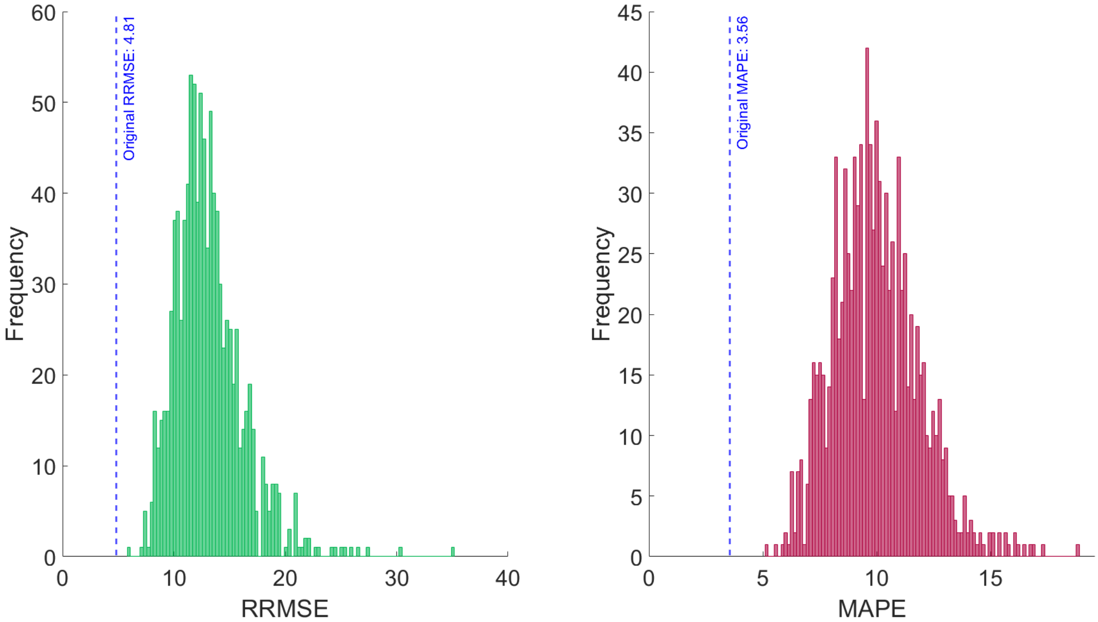


Figure 36: Distribution of RRMSE and MAPE based on predictions from the Gandalf model on 1000 simulated realizations from the Gandalf model. Prediction scheme 1 was used to generate the predictions, once for each simulation.

A closer inspection of the parameters for the Gandalf model could explain the above result. Since the predictions were generated using prediction scheme 1, the Gandalf model was fitted 28 times for each of the 1000 simulated realizations. For the 200 first simulations, the parameter estimates were plotted together with a dotted line where the parameter estimates on the training data were. The confidence interval are also based on the original data; they are therefore constant over time. Figure 37 displays the result. The estimates are mostly contained inside the confidence intervals of the original estimates. Some bias can be observed for α_1 and β_1. This supports the fact that the simulations were generated correctly, since they suggest the same model as the one originally fitted.

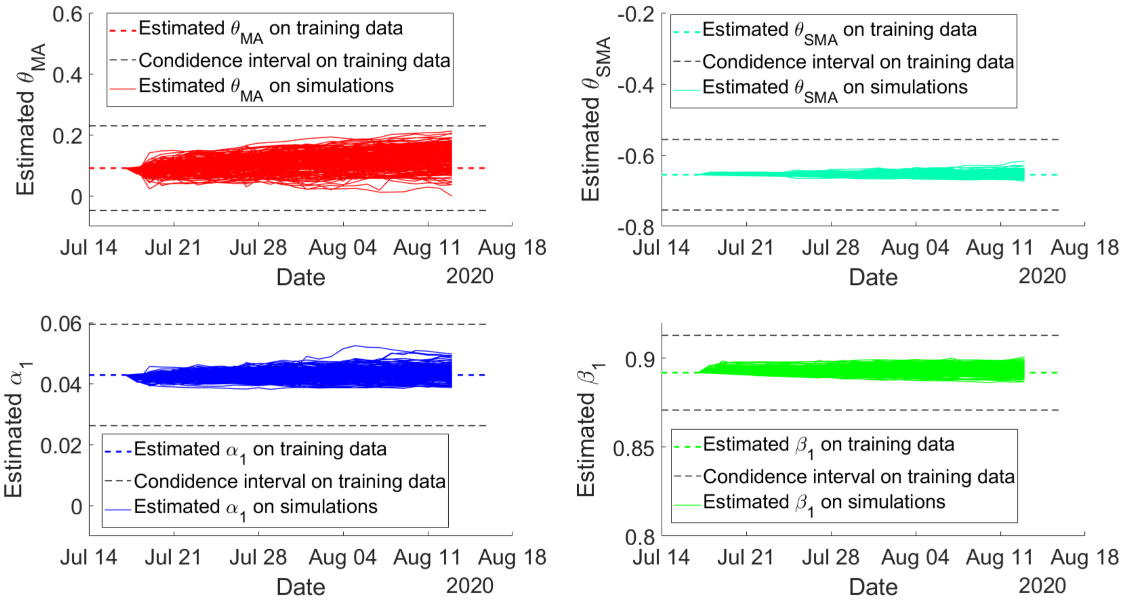


Figure 37: The deviation of the four Gandalf parameters on the 200 simulated realizations. For each simulated realization, prediction scheme 1 was used to fit models and predict one day at a time. This gives 28 fitted models and parameter estimates for each simulation.

The Gandalf model is fitted and predicts the new cases on log-scale. Hence, it was hypothesised that the transformation back to normal scale caused the above result. The log-transformed prediction of the simulated data was evaluated, and the results are shown in Figure 38. Once again, all accuracy measures are above the original score.

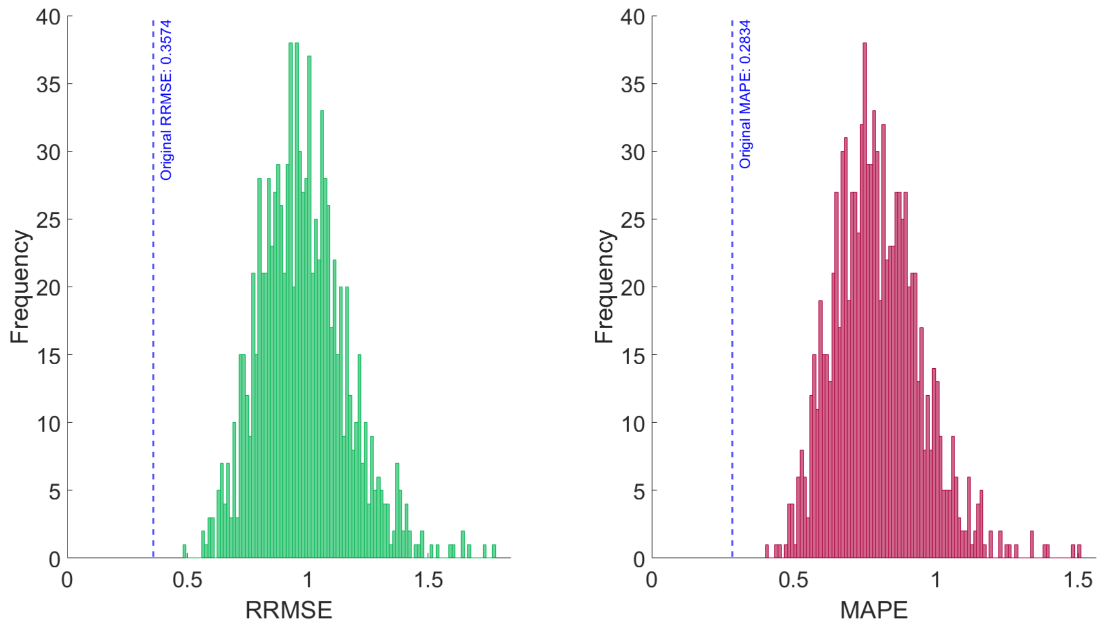


Figure 38: Distribution of RRMSE and MAPE based on log-scaled predictions from the Gandalf model using simulated realizations from the Gandalf model on log scale. Prediction scheme 1 was used to generate the predictions, once for each of the 1000 realizations.

Even though the accuracy of the Gandalf model on the Gandalf simulations were worse than on the actual test data, Figure 36 can serve as a baseline for the two other models. The SARIMA model was expected to predict the results quite well, as the model is closely related to the Gandalf model. By inspection of Figure 39, this was the case.

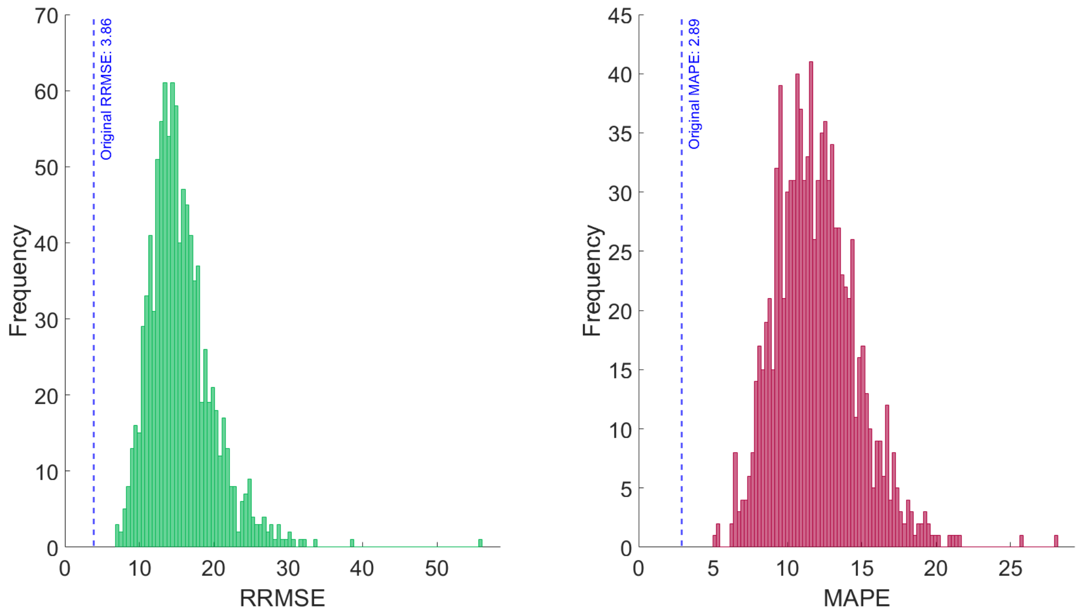


Figure 39: Distribution of RRMSE and MAPE based on predictions from the SARIMA model on 1000 simulated realizations from the Gandalf model. Prediction scheme 1 was used to generate the predictions, once for each realization.

The main reason for conducting the simulation study in the first place was to see how well the CNN-LSTM model predicted the simulated realizations. Again, there is no analytical expression for the prediction error for the machine learning model. Assuming the Gandalf model is a perfect model for the Covid-19 time series, this was another way to monitor the spread of the model's prediction. Observe that most of the RRMSE scores for the CNN-LSTM model around 40. Once again, none of the forecasts were more accurate on the simulated realizations than the actual test data w.r.t. the two measures.

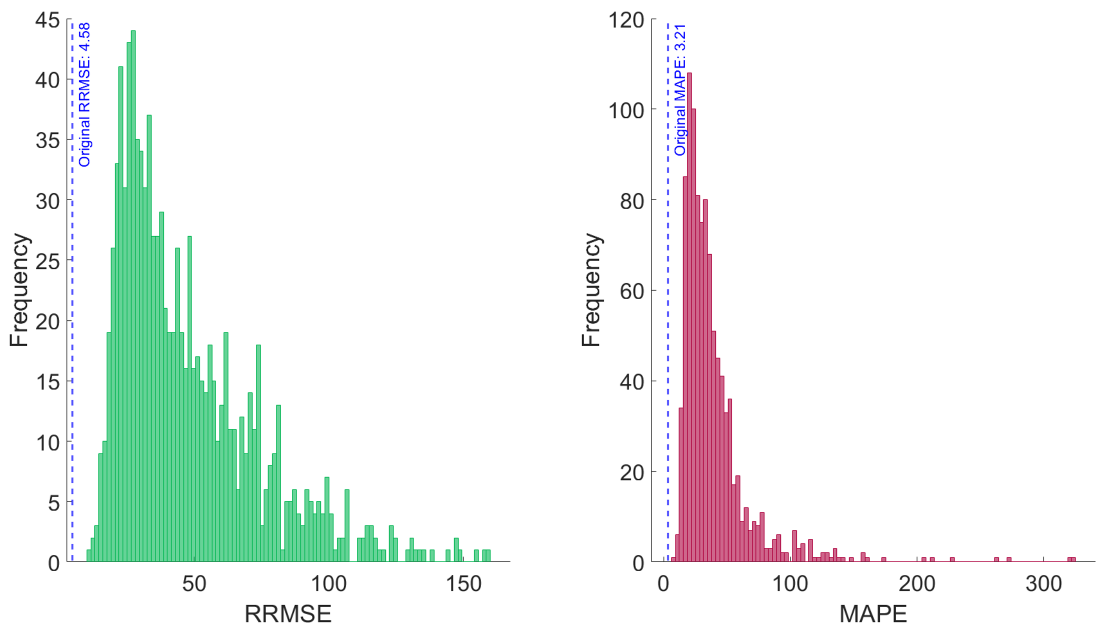


Figure 40: Distribution of RRMSE and MAPE based on predictions from the CNN-LSTM model on 1000 simulated realizations from the Gandalf model. Prediction scheme 1 was used to generate the predictions, once for each realization. The mean CNN-LSTM prediction was used to generate the forecasts.

To make the comparisons easier, it was helpful to have concrete measures to summarize the three pairs of histograms. The arithmetic mean gives precise information about how accurate each model was in total, while the standard deviation (Std) measures the spread of the distributions. However, for the purposes of this study, the measure of spread of each distribution should be relative to the magnitude of the errors. **The relative standard deviation** (RStd) satisfies these criteria. It is calculated by dividing the standard deviation of the distributions by their mean. Table 4 displays the RStd for both distributions of measures, for each model. The Gandalf model has the lowest relative spread for both measures, and the lowest mean. The SARIMA model accuracy measures are only slightly more spread out. The RStd of the RRMSE scores for the mean CNN-LSTM predictions are about twice as high as the other two, while it was four times higher for the MAPE scores.

Table 4: Relative standard deviation of accuracy measures of 28-day forecasts on 1000 simulated realizations from the Gandalf model. The models were initially trained on Zain's data set.

Models	RRMSE			MAPE		
	Mean	Std	RStd	Mean	Std	RStd
Gandalf	13.08	3.10	0.24	9.97	1.89	0.19
SARIMA	15.61	4.42	0.28	11.96	2.76	0.23
CNN-LSTM	47.57	26.24	0.55	39.48	30.12	0.76

5 Discussion

This section discusses the results from the previous section, suggests possible improvements, and addresses some problems with the applicability of the models.

5.1 The performance of the three models

The previous section studied the performance of the three models across the data sets, which are now discussed.

5.1.1 Accuracy

The models were developed for the early parts of the Covid-19 data sets, where they all delivered fairly accurate forecasts overall. Across the two yeas of the data set, a lot has happened to the way a positive Covid-19 test is treated. Everything from the different variations of the virus to the rapid antigen tests and the vaccines have contributed to this. A worry was that this might alter the behaviour of the time series, and thus hamper the performance of the models. As this thesis have shown, this was not the case; the models continued to deliver good forecast across both time series (in the sense that the mean RRMSE for all models were around 20).

The SARIMA model gave the most accurate forecasts overall. However, the gap between the SARIMA model and the Gandalf model was narrow when comparing MAPE scores. The prediction interval for the Gandalf model were consistently much narrower than the SARIMA model's. Thus, an argument can be made for choosing either of these models. The CNN-LSTM model has the overall highest RRMSE and MAPE score and does not come with a prediction interval. This is major drawback of the machine learning model. It is still quite fascinating that the CNN-LSTM manages to predict with such good accuracies, since it includes more than 300000 parameters: especially for the lower sample sizes.

One of the largest gaps between the models was illuminated when the RRMSE scores were grouped by forecast length (and prediction scheme, since all 28-day forecasts used prediction scheme 1 and vice versa). The SARIMA model performed much better than the other models on 28-day forecasts. On the seven-day predictions, the models performed equally good. Moreover, the SARIMA model was the only model that performed worse on these forecasts. This has a natural explanation, that will be discussed later. On both measures of accuracy, the machine learning model gave the worse performance on the 28-day forecasts, while it was just as good at forecasting seven days with prediction scheme 2. A possible flaw was that the CNN-LSTM models were only fitted initially for the 28-day predictions with predictions scheme 1. On the other hand, the SARIMA models were fitted at each step. The decision was based on the fact that the CNN-LSTM models had a much longer time to fit (see Table 2). This seems to have had a clear effect on the measures of accuracy in Figure 33 and Figure 43. These high inaccuracies of the 28-day forecast translates over to the other comparisons in Figure 33 and Figure 43. In cases where only 50 training data were available, this effect was quite clear, since the machine learning model has to forecast half the length of the training data. If the CNN-LSTM model were forecasted in the same way, the accuracies on the 28-day forecasts might have been more comparable to the SARIMA model, and thus, the other comparisons would also go more in the favour of the CNN-LSTM model.

It is important to note that the above statements are only based on the 28 partitions of each data set in Table 3. The results might have changed if the analysis were done with different ending dates, with a different set of sample sizes, and with other prediction lengths.

Overall, the models were more precise on the global data set. This data set has a much higher volume of new cases, and the relative changes are not as great as in the Norwegian data set. This means than the RRMSE and MAPE often will be lower for the global data set. This was despite the fact that some lack of fit was observed for the SARIMA model. Ideally, these relative measures of accuracy should be comparable across the two data sets. In practice, this remark shows that these comparisons may be flawed.

5.1.2 The changing volatility and the effect of sample reduction

Figure 12 showed that the series $(1-B)(1-B^7)X_t$ exhibits conditional heteroscedasticity, with higher variations in the early parts of both data sets. This suggests that a conditional variance model should be used for all SARIMA models when the earliest parts of the data sets were included. This might explain why the Gandalf model performed better than the SARIMA model on Taraldsen's data set. At the later periods, the series $(1-B)(1-B^7)X_t$ looks quite stationary for both data sets. Thus, a GARCH$(1,1)$ model should have little to no advantage here; it might even cause overfitting. For lower sample sizes, there is a greater chance of avoiding the heteroscedastic parts of the data sets. On the other hand, if all previous data is used, these parts are sure to be included. This is reflected in Table 3 and the associated Figure 33 and Figure 43, where the SARIMA model is more accurate for lower sample sizes. The opposite relation is true for the Gandalf model and the CNN-LSTM model.

The estimated θ_{MA} and θ_{SMA} were stable for both the SARIMA models on both original data sets, seen Figure 21, Figure 22 and Figure 27, with the exception of θ_{MA} for the Gandalf model on the global data set. Together with Figure 12, this suggests that the moving average structure of these models fits the data well. Otherwise, they should vary slightly with each additional data point. The major differences between the parameters of the two models were in the estimates associated with the noise component. The same figures show significant declines in the estimated σ^2 parameters. The models were trained on the parts of the data with conditional heteroscedasticity (see Figure 12). Hence, the SARIMA model might have to reduce its estimated variance when fitted on the test data, which is more stable than the average variation of the training data. With prediction scheme 1, the models were fitted for each new day. The variance can therefore be adjusted to the new data before each new value is predicted. This is not the case when using prediction scheme 2, where the initial estimated σ^2 is used for the entire forecast. This may explain why the SARIMA model is the only model with increased RRMSE and MAPE on the seven-day forecasts. In Figure 21 and Figure 27, the noise estimates α_1 and β_1 in the Gandalf model, are more stable for the two original data sets. However, if this were to be a problem, Figure 28 should then be the SARIMA model's downfall, since the parameter estimates were never updated on a 28-day forecast. This was not the case, which suggests that the persistence volatility does not cause any significant drawback to the performance of the SARIMA model for Zain's data set. Additionally, the SARIMA model performs better than the Gandalf model on the seven-day forecast trained on Zain's data set. An explanation might be that the end of this data set is far enough away from the part with higher variation. There is some evidence of the Gandalf model achieving more accurate seven-day forecasts when the elevated variance is closer to the end of the training data, where the SARIMA model has not had time to compensate. A closer investigation of the performances of each model should be made around these problem areas.

The downwards trajectory of the σ^2 estimates for the SARIMA model on the two original data sets explains the narrowing prediction intervals for 28-day forecasts in Figure 20 and Figure 26. The earlier assessment that the Gandalf model have narrower prediction intervals might not be true for later data partitions, where the time series are close to homoscedastic and has less variation overall.

Figure 30 and Figure 29 suggest that the similarity between the forecast of the two models increases as the sample size reduces. A close inspection of Figure 12 gives a more reasonable interpretation. In the figure, the part of the data with increased volatility is excluded from the training set when only considering the past 50 observations, and mostly excluded with the most recent 100 observations. Thus, the added GARCH$(1,1)$ process should not differ much from the regular white noise process. Figure 31 supports this statement, since the estimated variance for the SARIMA model stays constant across the 28-day forecast. Additionally, the estimated α_1 and β_1 stayed close to 0 and 1, respectively. This gives the following conditional variance process for the Gandalf model:

$$\sigma_t^2 = \alpha_0 + \alpha_1 Z_{t-1}^2 + \beta_1 \sigma_{t-1}^2 \approx \sigma_{t-1}^2,$$

since α_0 was fixed at 0.001. Thus, the GARCH1, 1) model approximates conditional homoscedasticity. If α_0 had not been fixed, an alternative set of estimates would have been that $\alpha_1 = \beta_1 \approx 0$ and $\alpha_0 \approx \sigma^2$. The similarity between the estimated θ_{MA} and θ_{SMA} in the two models supports the fact that the variance models are similar. This explains why the models' forecasts and prediction intervals align perfectly in Figure 30.

Even if the models preformed great overall, there were some partitions of the data sets that were seemingly harder for the models to model. This might simply be due to chance, but when all models preformed badly, it is reasonable to assume that the time series in question changed behaviour on those partitions. Maybe the vaccines started to become available to the masses. Or maybe a new variant increased in prominence. These connections are interesting in themselves and should be investigated.

5.1.3 Results from the simulation study

The main motivation behind the simulation study was to quantify the spread of the predictions from the CNN-LSTM model. None of the RRMSE scores nor the MAPE scores from the predictions on the simulated realizations were lower than they were on original test data, with the Gandalf model. This was strange since all the simulated realizations came from the Gandalf model. Most of the predictions on these were therefore expected to be more accurate than on the original test data. An initial hypothesis was that this was due to the transformation back to normal scale. After all, the SARIMA models were built to perform on the log-scaled data sets. However, the same phenomenon was observed. One cannot exclude the possibility of this being due to a programming error. However, the simulated prediction interval aligns quite well with the Figure 34, and the parameter estimates from the Gandalf model do not go astray. These results provide some assurance that the simulated realizations and the associated forecasts were generated correctly, which gives credibility to later findings.

By inspection of Figure 34, an alternative explanation could be that the actual test data behaves nicer than the simulated realizations, since these are observably much wilder. The high volatility in the beginning of the data set might partially explain this. If this is the correct explanation, comparing the spread of these predictions might still be fruitful. The SARIMA model is closely related to the Gandalf model and was expected to be about as accurate on the simulated realizations. However, the SARIMA model showed a lot wider prediction intervals on the forecasts ensuing Zain's data set. Hence, it was reasonable to assume a wider spread of the RRMSE scores for the SARIMA model than the Gandalf model. For the Gandalf model, the mean RRMSE score was 13.08, and the relative standard deviation was 0.24. On the MAPE scores, the mean and the RStd were slightly lower. The SARIMA model achieved almost as good results but has a slightly higher mean and a wider spread for both measures. However, the relative spread of this forecast is nowhere close the relative width of the prediction intervals on the actual test data, where the prediction interval for the SARIMA model was consistently three to four times wider than the Gandalf model's interval. Whatever spread is measured by the RStd on these measures of accuracy, it should not be compared directly to the prediction intervals. The machine learning models struggled to predict the simulated realizations, with a mean RRMSE around 50 and MAPE around 40. The RStd was two and four times higher than the Gandalf model, respectively, suggesting that the spread of its forecast on this partition of the data is higher.

5.2 Possible improvements

The models used in this thesis were, for the most part, replicated from the articles by Taraldsen and Zain et. al. While the models performed fairly well on both data sets, there are several plausible ways to improve the models.

5.2.1 The machine learning model

CNN-LSTM models has a lot of hyper parameters that might have a big impact on the results. Zain et. al. did their own hyper parameter tuning with the Optuna framework. Sadly, they did not publish all their optimal hyper parameters. Two of these parameters were essential to know when building the model, namely the dropout rate and the number of neurons in the fully connected layer. As noted earlier, the global data set used in this thesis was not identical to what Zain et. al. used. Thus, even the published hyper parameters might not be optimal. The same parameters were used across all partitions of the global data set. The same hyper parameters were also used for the Norwegian data set, where they are even less likely to be optimal. This was intentional, since if the models managed both data sets well using the same architecture, they effectively capture some ground truth about the spread of Covid-19. However, to increase the performance of the CNN-LSTM models, their hyper parameters model should be individually tuned for the Norwegian and the global data set. Ideally, they should be re-tuned for each evaluated partition of the two data sets.

Alterations to the entire model structure should also be included in the hyper parameter searches. This includes the number of layers of each type (CNN layers, LSTM layers and fully connected layers, pooling layers, and dropout layers) and the number of neurons in each of these. This would of course result in a lot longer training times, but if they do not take several days, this should not be a hindrance in practice.

Using the MSE as loss function might not be ideal for this problem. After all, the forecasts were evaluated based on the RRMSE and the MAPE. Some experimentation with the use of MAPE as loss function was made. Ironically, this resulted in a worse MAPE of the forecasts. In hindsight, this loss function might have required a completely different set of hyper parameters to give optimal forecasts. To choose the optimal number of epochs, the models should incorporate early stopping, e.g., with a stopping criterion set to 0.0015. Lastly, the median might be a better choice of measure of central tendency for the model's forecasts, as it deals better with outliers than the mean.

Figure 28 demonstrates a weakness of the CNN-LSTM model in its current form: it does not capture the weekly variation and the models fades out as the prediction length increases. This did not reduce the performance of the model in this one example, but this might not be consistent on other partitions of the data sets. This motivates a variation of the model: The weekday model. This model is really seven sub models, one for each day of the week. Thus, one of these models only considers the new cases on each Monday, another considers all Tuesdays, and so on. Together, they can be used to predict the new cases on each ensuing day. This could potentially be a better alternative, as it would capture the weekly structure by construction. A problem with this proposed model is that this reduces the amount of training data for each of these sub models by a factor of seven. The model was tested (and the code for the model are included in the Python-file in Appendix B), but the initial tests did not achieve great accuracies, which is why it was excluded from the thesis. Looking back, each sub model should be treated individually, with potentially unique model architectures (determined by a unique hyper parameter search).

5.2.2 The SARIMA models

The structure of the SARIMA models was natural for modeling the Covid-19 new cases data, as both data sets have a clear weekly pattern and an ever-changing mean. This is supported by the highly desirable behaviour of the residuals on Taraldsen's data set in Figure 14. Figure 12 showed that the series $(1 - B)(1 - B^7)X_t$ exhibits higher-than-usual variance in the early parts of this data sets. Together with the ACF plot of the squared residuals, this suggests that a conditional variance model would have a better fit to the noise than a white noise model. Even if the residuals and the accuracy of the two forecasts on Taraldsen's data set improved with the added GARCH$(1, 1)$ model, the squared residuals on this data set was not improved by the added GARCH$(1, 1)$ model. It is possible that a different GARCH model might have been a better alternative. This could be determined by a grid search for p and q, where each combination (up to some limit) was compared based on the AICc. This must be done with caution. As should be clear by now, the SARIMA model fits the later partitions of the data set better than the Gandalf model. Even the GARCH$(1, 1)$ model seem to cause overfitting, as the predictions are slightly less accurate overall. The ACF plot of the residuals based on Zain's data set from the SARIMA model showed that many lags were significant. A similar grid search as described above might have improved the results on the global data set even further and should have been attempted.

As described in Section 2.4, Matlab uses a conditional likelihood to estimate the parameters for the SARIMA models. The exact likelihood should give slightly more precise estimates, especially for the reduced sample sizes. Another suggestion for lower sample sizes is to assume that either Z_t (or e_t, when a GARCH model is used) follows Student's t-distribution instead of the Gaussian distribution.

In all plots of the parameters for the Gandalf model (see Figure 21, Figure 27 and Figure 37) $\alpha_1 + \beta_1$ was consistently quite close to 1. The Exponential GARCH$(1, 1)$ model fixates this sum to be one, and should be considered as the noise process for these data sets.

None of the models presented in this thesis takes into account that the new cases are integers. In an earlier article, Taraldsen (2020a) gives several alternatives for modeling new Covid-19 cases. Interer-valued GARCH (INGARCH) models are one such example. The COM-Poisson process has been shown to deal well with both overdispersion and underdispersion, which are commonalities when modeling count data. Zhu (2012) proposes a COM-Poisson INGARCH model for modeling time series of counts. This model should be tested on both data sets used in this thesis.

5.3 A note on the application of the models in the real world

For the most part, all three models manage to forecast the daily new Covid-19 cases data set fairly well. However, the utility of these forecasts can be questioned. The new cases data only includes reported cases. The regular weekly fluctuations of the time series reflect this, with a drastic decrease in the reported cases at the weekends. These are not correct representations of the spread of the virus. Neither are the forecasts of these series. How useful is this if the goal is to foresee the next outbreak? For this to bring utility, multiple data sources should probably be used in conjunction. Examples of this are mobility data and more detailed models of the person-to-person trends. However, each data source introduces more uncertainty in the final prediction. The models might be used on top of other types of models to account for the weekly fluctuations in the new cases data sets. There might also be more utility in creating models based on the weekly new cases, as these gives a better representation of the actual number of infected people. The hospitalizations due to Covid-19 might also paint a more correct picture of the severity of the situation, especially in the post-vaccine era, where most infected people only have minor symptoms. The problem with both of these alternatives is that they contain less samples than the daily new-cases data set.

The different partitions of the data sets might require different models for optimal results. Thus, the models should continually be tested on the newest data if they were to be used in a real-world setting.

6 Conclusion

This thesis presented promising results for the three models. Overall, the SARIMA model performed better than the other two models. For the CNN-LSTM model, this might be due to sub-optimal hyper parameters. There seems to be persistence of volatility in the beginning of both the Norwegian and the global data set. On these partitions, the Gandalf model is more accurate than the SARIMA model and has a much tighter prediction interval. On most of both data sets, however, the variance is homoscedastic and much lower in magnitude. Here, the added GARCH effect merely introduces more uncertainty, possibly resulting in overfitting. Additionally, the reduced variance estimate will reduce the width of the prediction interval for the SARIMA model on the majority of both data sets. The simulation study showed that the spread of the two SARIMA models was about as low, while the spread of the machine learning model was multiple times higher. However, the simulated realizations where wilder than the test data, so the results are not directly comparable to the results on this data set. While the CNN-LSTM model competes with the other models, there were few scenarios where it achieved better results. It is still quite remarkable that a model with over 300000 parameters gives accurate forecasts for such low sample sizes. If one had to choose one of these models for modeling daily Covid-19 new cases, the SARIMA model with Gaussian white noise would most likely give the best results overall. However, if clear signs of conditional heteroscedasticity are present at during the last few days of the training data, the Gaussian white noise may be replaced by GARCH$(1, 1)$ noise.

References

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, *31*, 307–327.

Brockwell, P. J. & Davis, R. A. (2016). *Introduction to time series and forecasting*. Springer.

Dehning, J., Zierenberg, J., Spitzner, F., Wibral, M., Neto, J., Wilczek, M. & Priesemann, V. (2020). Predictions, role of interventions and effects of a historic national lockdown in india's response to the covid-19 pandemic: Data science call to arms. *Harvard data science review*. https://doi.org/https://doi.org/10.1162/99608f92.60e08ed5

Ekinci, A. (2021). Modelling and forecasting of growth rate of new covid-19 cases in top nine affected countries: Considering conditional variance and asymmetric effect. *Chaos, Solitons Fractals*, *151*, 111227. https://doi.org/https://doi.org/10.1016/j.chaos.2021.111227

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *The Econometric Society*, *50*, 987–1007.

Hamilton, J. D. (1994). *Time series analysis*. Princeton University Press.

Higham, N. J. (2009). Cholesky factorization. *WIREs computational statistics*, *1*, 251–254. https://doi.org/https://doi.org/10.1002/wics.18

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M. & Inman, D. J. (2021). 1d convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, *151*, 107398. https://doi.org/https://doi.org/10.1016/j.ymssp.2020.107398

Kumar, K. A., Kalaga, D. V., Sai Kumar, C. M., Chilkoor, G., Kawaji, M. & Brenza, T. M. (2021). Forecasting the dynamics of cumulative covid-19 cases (confirmed, recovered and deaths) for top-16 countries using statistical machine learning models: Auto-regressive integrated moving average (arima) and seasonal auto-regressive integrated moving average (sarima). *Applied Soft Computing*, *103*, 107161. https://doi.org/https://doi.org/10.1016/j.asoc.2021.107161

Li, M.-F., Tang, X.-P., Wu, W. & Liu, H.-B. (2013). General models for estimating daily global solar radiation for different solar radiation zones in mainland china. *Energy Conversion and Management*, *70*, 139–148. https://doi.org/https://doi.org/10.1016/j.enconman.2013.03.004

Luan, Y. & Lin, S. (2019). Research on text classification based on cnn and lstm. *2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, 352–355. https://doi.org/10.1109/ICAICA.2019.8873454

MachineLearningMastery. (n.d.). *Lstm model architecture for rare event time series forecasting*. https://machinelearningmastery.com/lstm-model-architecture-for-rare-event-time-series-forecasting/

Pascanu, R., Mikolov, T. & Bengio, Y. (2012). Understanding the exploding gradient problem. *CoRR*, *abs/1211.5063*. http://arxiv.org/abs/1211.5063

Peltarion. (2022, May 20). *Max pooling block 1d*. https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/max-pooling-block-1d

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Taraldsen, G. (2020a). A toy model for covid-19 data. *Researchgate*.

Taraldsen, G. (2020b). A wizard predicts new covid-19 cases. *Researchgate*.

TensorFlow. (2022a, May 20). *Tf.keras.layers.dropout*. https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout

TensorFlow. (2022b, May 29). *Tf.keras.layers.lstm*. https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM

TensorFlow. (2022c, May 20). *Tf.keras.optimizers.adam*. https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam

The MathWorks, I. (2022a, May 13). *Maximum likelihood estimation for conditional mean models*. https://se.mathworks.com/help/econ/arima-maximum-likelihood-estimation.html

The MathWorks, I. (2022b, May 13). *Mmse forecasting of conditional mean models*. https://se.mathworks.com/help/econ/mmse-forecasting-for-arima-models.html

The MathWorks, I. (2022c, May 13). *Model comparison tests*. https://se.mathworks.com/help/econ/model-comparison-tests.html#btb4y0n

The MathWorks, I. (2022d, May 19). *Simulate stationary processes*. https://se.mathworks.com/help/econ/simulate-stationary-arma-processes.html

The MathWorks, I. (2022e, May 19). *Simulate stationary processes*. https://se.mathworks.com/help/econ/engles-arch-test.html

TowardsDataScience. (n.d.). *Types of convolution kernels : Simplified*. Retrieved, from https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37

Varsamopoulos, S., Bertels, K. & Almudever, C. (2018). *Designing neural network based decoders for surface codes*.

Wei Zhang, Gaoliang Peng & Chuanhao Li. (2017). Bearings fault diagnosis based on convolutional neural networks with 2-d representation of vibration signals as input. *MATEC Web Conf.*, *95*, 13001. https://doi.org/10.1051/matecconf/20179513001

WHO. (2022, February 20). *Who coronavirus (covid-19) dashboard*. https://covid19.who.int/data

Zain, Z. M. & Alturki, N. M. (2021). On the true number of covid-19 infections: Effect of sensitivity, specificity and number of tests on prevalence ratio estimation. *Journal of Control Science and Engineering*, *2021*.

Zhang, J., Peng, Y., Ren, B. & Li, T. (2021). Pm2.5 concentration prediction based on cnn-bilstm and attention mechanism. *Algorithms*, *14*, 208. https://doi.org/10.3390/a14070208

Zhang, W., Huang, G., Wang, G. & Wang, Y. (2019). Prediction high frequency parameters based on neural network. *IOP Conference Series: Materials Science and Engineering*, *631*, 052035.

Zhang, Z. (2016). Derivation of backpropagation in convolutional neural network (cnn). *University of Tennessee, Knoxvill*.

Zhu, F. (2012). Modeling time series of counts with com-poisson ingarch models. *Mathematical and Computer Modelling*, *56*(9), 191–203. https://doi.org/https://doi.org/10.1016/j.mcm.2011.11.069

Appendix

A Figures and tables

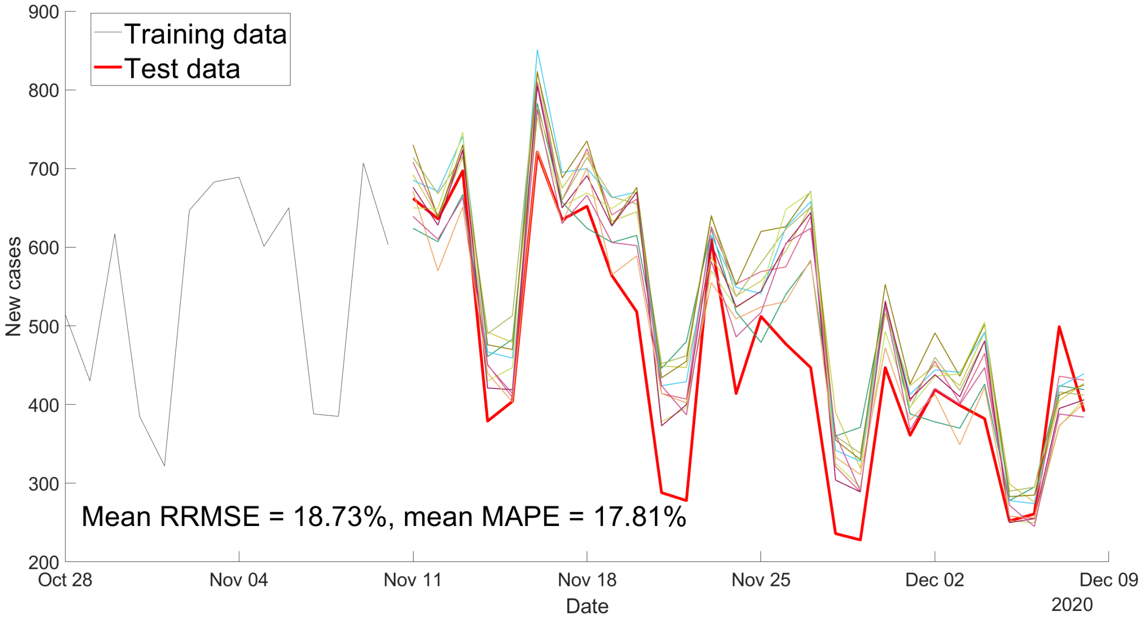


Figure 41: Forecast with ten CNN-LSTM models from November 11th to December 8th 2020 on the Norwegian new-cases data. The previous 264 days there used as training data.

Table 5: All forecast results from the mean prediction of ten CNN-LSTM models and the mean RRMSE of the ten (individual) models. All 28 day predictions used prediction scheme 1, while prediction scheme 2 was used for all seven day predictions.

Data set	Last training date	Sample size	Days ahead prediction	CNN-LSTM mean prediction		Ten CNN-LSTM models	
				RRMSE	MAPE	Mean RRMSE	Mean MAPE
Norway	30.05 2020	100	7	64.23	88.08	70.75	92.84
			28	73.05	87.49	75.77	89.19
		50	7	64.52	40.73	65.04	42.56
			28	52.72	41.21	53.58	41.95
	10.11 2020	264	7	8.03	7.91	12.68	12.36
			28	17.91	16.96	18.73	17.81
		100	7	10.63	10.73	11.16	11.26
			28	25.01	25.98	25.48	26.07
		50	7	11.37	11.27	11.97	11.76
			28	27.71	28.36	28.05	28.45
	16.12 2020	300	7	14.18	11.83	15.43	13.45
			28	28.85	22.54	29.26	22.76
		100	7	7.41	6.65	12.72	12.38
			28	29.22	22.85	30.24	23.82
		50	7	18.28	14.00	18.35	14.23
			28	25.81	20.82	26.17	20.98
	04.07 2021	500	7	7.85	7.72	11.23	10.38
			28	17.06	11.72	17.57	12.66
		100	7	11.56	11.95	13.11	13.03
			28	30.38	20.10	30.78	21.08
		50	7	10.79	9.23	11.41	9.40
			28	35.02	23.32	35.16	23.42
	16.01 2022	696	7	23.05	21.81	23.17	21.81
			28	18.76	16.57	20.72	18.27
		100	7	24.04	18.24	25.53	21.51
			28	29.56	30.89	29.89	31.11
		50	7	23.76	22.69	24.88	23.65
			28	45.55	46.06	45.72	46.15
Global	12.05 2020	100	7	10.72	9.85	10.79	9.89
			28	7.34	6.33	7.54	6.44
		50	7	14.38	13.47	14.41	13.47
			28	9.86	8.51	9.93	8.56
	17.07 2020	196	7	7.00	4.18	7.16	4.51
			28	4.58	3.21	4.86	3.59
		100	7	5.90	3.59	6.17	3.93
			28	4.17	3.20	4.46	3.41
		50	7	4.90	3.04	5.03	3.93
			28	3.79	3.12	4.21	3.31
	29.10 2020	300	7	11.19	9.90	11.27	9.94
			28	12.19	11.07	12.28	11.14
		100	7	7.97	6.79	8.05	7.02
			28	9.03	7.59	9.13	7.71
		50	7	9.44	8.41	9.49	8.46
			28	10.46	9.34	10.53	9.39
	17.05 2021	500	7	13.36	12.99	13.42	12.99
			28	9.29	7.89	9.43	8.19
		100	7	20.43	20.33	20.46	20.33
			28	15.30	13.83	15.36	13.87
		50	7	18.24	17.88	18.26	17.88
			28	38.96	39.08	39.03	39.08
	14.01 2022	743	7	12.17	12.08	12.80	12.32
			28	20.59	20.08	21.08	20.42
		100	7	10.35	10.30	10.58	10.40
			28	20.64	20.11	20.75	20.22
		50	7	11.38	10.64	11.76	11.08
			28	85.18	82.83	85.18	82.83

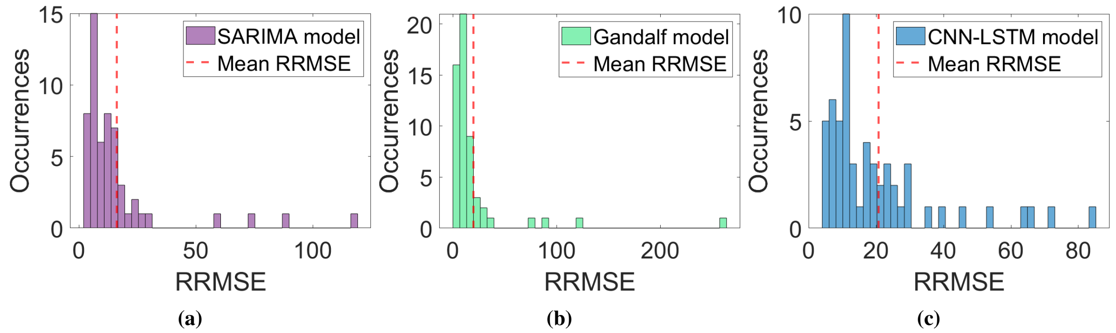


Figure 42: RRMSE scores for each model from Table 3.

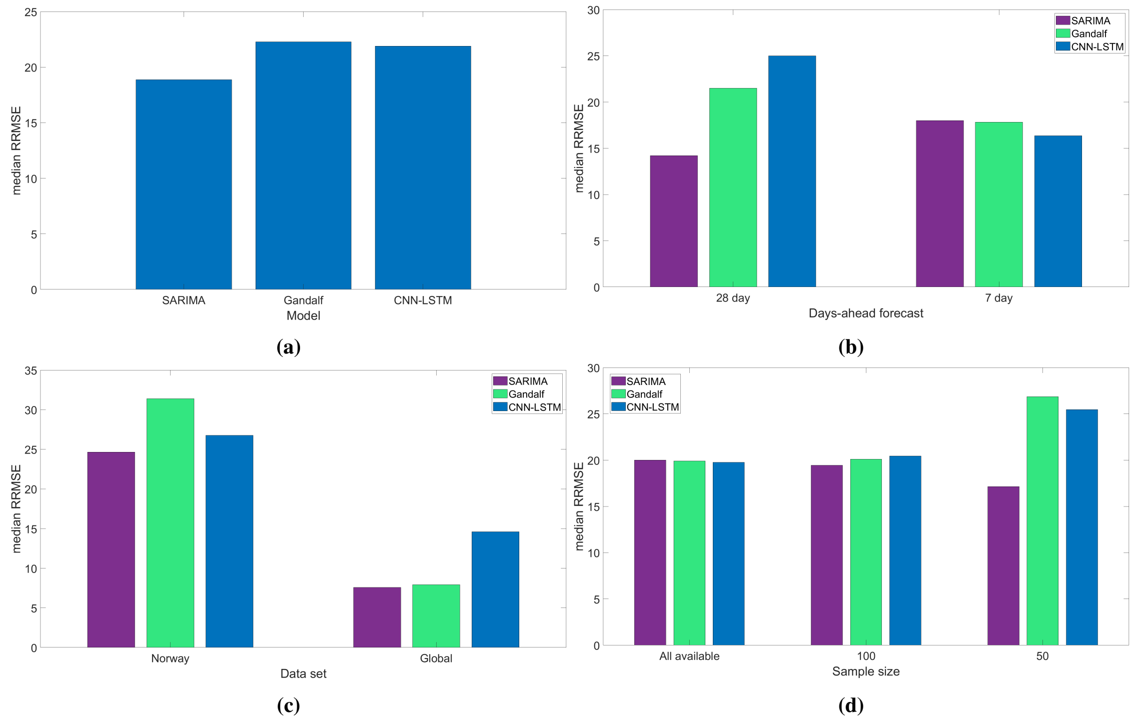


Figure 43: Comparison of models based on RRMSE scores in Table 3

B Code

In this thesis, three programming languages were utilized. Matlab was used for the econometric model and for all the generated plots, while Python was used for the CNN-LSTM model. Finally, R was used for pre-processing of the Norwegian and the global data set. To plot the results from the CNN-LSTM model, the forecasts made in Python was saved and loaded in Matlab. It should be noted that most of the functionality is commented out. To run the code, simply uncomment the desired section of code. Additionally, the file paths should be replaced to fit the user's computer. The code files and the two data sets are available on request.

R-code

The following code was used to pre-process the global and the Norwegian data new cases data sets.

```
# Pre-process global
df = read.csv("WHO-COVID-19-global-data.csv")
df = df[, c(1, 2, 5)]
colnames(df) = c("Date","Country", "New_cases")
# Cut-off at February 20th 2022
df$Date = as.Date(df$Date , format = "%Y-%m-%d")
last_date = as.Date("2022-02-20", format = "%Y-%m-%d")
df$Date_diff = last_date - df$Date
df_ = df[df$Date_diff ≥ 0,]

new_cases_one_country = df_[df_$Country == "AF",]
worldwide_aggregated = new_cases_one_country[
  !is.na(new_cases_one_country$Country),3]

countries_list = unique(df[df$Country!= "AF",]$Country)
countries_list = countries_list[!is.na(countries_list)]

for (country in countries_list){
  new_cases_one_country = df_[df_$Country == country,]
  worldwide_aggregated = worldwide_aggregated + new_cases_one_country[
    !is.na(new_cases_one_country$Country),3]
}
write.csv(worldwide_aggregated[-1],"worldwide_aggregated_new.csv",
          row.names = FALSE)

# Pre-process Norway
df = read.csv("antall-meldte-covid-19-t.csv", sep = ";")
df = df[, c(1, 3)]
colnames(df) = c("Date", "New_cases")
# Cut-off at February 20th 2022
df$Date = as.Date(df$Date , format = "%d.%m.%Y")
last_date = as.Date("2022-02-20", format = "%Y-%m-%d")
df$Date_diff = last_date - df$Date
df_ = df[df$Date_diff ≥ 0,]
write.csv(df$Nye.tilfeller,"new_cases_Norway.csv", row.names = FALSE)
```

Python-code

The following code was used to derive all predictions from the CNN-LSTM model. The results were then transferred to the Matlab file and was from there plotted.

```python
import random
import time
from numpy import array
import numpy as np

import pandas as pd
import os
import tensorflow as tf
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers import RepeatVector
from keras.layers import TimeDistributed
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt


seed_value = 1234

# Set seed in all environments to help reproducibility
def set_all_seeds(seed_value):
    # 1. Set `PYTHONHASHSEED` environment variable at a fixed value
    os.environ['PYTHONHASHSEED']=str(seed_value)
    # 2. Set `python` built-in pseudo-random generator at a fixed value
    random.seed(seed_value)
    # 3. Set `numpy` pseudo-random generator at a fixed value
    np.random.seed(seed_value)
    # 4. Set `tensorflow` pseudo-random generator at a fixed value
    tf.random.set_seed(seed_value)


# import data

# Set current working directory one layer up
path_parent = os.path.dirname(os.getcwd())
os.chdir(path_parent)

ts_global = np.array(pd.read_csv(os.getcwd() + r"\Data\new_cases_global.csv"))
ts_norway = np.array(pd.read_csv(os.getcwd() + r"\Data\new_cases_Norway.csv"))


def split_sequence(sequence, n_steps):
    # Formats the data for supervised learning
    X, y = list(), list()
    for i in range(len(sequence)):
        # find the end of this pattern
        end_ix = i + n_steps
        # check if we are beyond the sequence
        if end_ix > len(sequence ) -1:
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return array(X), array(y)

def k_step_prediction(k, x_train, model, scaler, prev_days = 7):
    '''
    Iteratively performes a one step prediction and use the newly
    predicted as input to forecast the next day
    :param k: amount of days ahead to predict
    :param x_train: the time series used
```

```
    :param model: pre-trained model
    :param scaler: the pre-trained min-max scaler,
    used to revert the prediction back to normal scale
    :param prev_days: Amount of previous days used to predict the next day.
    :return: k-step prediction
    '''
    predictions = []
    x_input = x_train[-prev_days:] # last prev_days days
    x_input = x_input.reshape((1, prev_days, 1))
    yhat = model.predict(x_input, verbose=0)[0][0].reshape(1,1)
    predictions.append(yhat)
    for i in range(1, k):
        x_input = x_input.reshape(prev_days, 1)
        # remove first day and add prediction to the end of the input
        x_input = np.concatenate((x_input[1:], yhat), axis = 0)
        x_input = x_input.reshape((1, prev_days, 1))
        yhat = model.predict(x_input, verbose=0)[0][0].reshape(1,1)
        predictions.append(yhat)

    for i in range(len(predictions)):
        predictions[i] = scaler.inverse_transform(predictions[i])[0][0]
    return predictions


def train_model(X, y, days_ahead, prev_days = 7):
    '''
    :param X: Data matrix
    :param y: Response vector
    :param days_ahead: days ahead to be forecasted after last input
    :param n_steps: amount of previous days used as input in forecasting
    :return:
    '''
    model = Sequential()
    model.add(Conv1D(filters=64, kernel_size=3, activation='relu', ...
        input_shape=(prev_days, 1)))
    model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
    model.add(MaxPooling1D(pool_size=1, strides=1))
    model.add(Dropout(0.2, seed=seed_value))
    model.add(Flatten())
    model.add(RepeatVector(1)) # days_ahead
    model.add(LSTM(200, return_sequences=True))
    model.add(TimeDistributed(Dense(50))) # 100
    model.add(TimeDistributed(Dense(1)))
    model.compile(optimizer='adam', loss='mse') # 'mean_absolute_percentage_error'
    # fit model
    history = model.fit(X, y, epochs=100, verbose=0, batch_size=22)  # skal v re ...
        472, 22
    loss = history.history['loss']
    # print(model.summary()) # uncomment to view amount of variables
    return model, loss


def prediction_global(ts, endpoint, prev_days, len_train = -1, days_ahead = 7):
    # The new version uses the repeat vector layer, and uses the
    # Time Distributed Wrapper at the end insted of at the beginning.

    '''
    :param ts: Input time series to be forecasted
    :param endpoint: Last index used for training the model
    :param prev_days: Amount of days used to predict the next days
    :param len_train: Amount of days back from endpoint used to train the model
    :param days_ahead: Amount of days forecasted after endpoint
    :return: days_ahead-forecast
    '''
    set_all_seeds(seed_value)
    if len_train == -1:
        len_train = endpoint
    scaler = MinMaxScaler()
    # only use train data  to fit scaler:
    x_train = ts[endpoint - len_train:endpoint].reshape(len_train, 1)
    scaler.fit(x_train)
    x_train = scaler.transform(x_train)
    X, y = split_sequence(x_train, prev_days)
```

```
    # reshape from [samples, timesteps] into
    # [samples, subsequences, timesteps, features]
    X = X.reshape((X.shape[0], prev_days, 1))

    preds = np.zeros((10, days_ahead))
    loss = []
    for i in range(10):
        start_time = time.time()
        model, loss_i = train_model(X, y, days_ahead, prev_days=prev_days)
        loss.append(loss_i)
        print("Time used to fit model", i+1, ": ", (time.time() - start_time), " ...
            sec")
        start_time = time.time()
        preds[i] = np.round_(k_step_prediction(
            days_ahead, x_train, model, scaler, prev_days=prev_days))
        print("Time used to forecast with model", i + 1, ": ",
              (time.time() - start_time), " sec")

    np.savetxt(os.getcwd() + r"\Predictions/glob_preds_" + str(days_ahead) +
               "_days_ahead_from_" + str(endpoint) + "_with_" + str(len_train) +
               "_train_data.csv", preds)

    # This part can be uncommented to inspect and save the mean loss function
    # loss = array(loss)
    # mean_loss = loss.mean(axis = 0)
    # print(mean_loss.shape)
    # np.savetxt(os.getcwd() +
    #              r"\Predictions/mean_loss_on_original_test_data.csv", mean_loss)
    # print(mean_loss.shape)
    # plt.plot(mean_loss)
    # plt.ylabel('loss')
    # plt.xlabel('epoch')
    # plt.show()
    return preds

def prediction_norway(ts, endpoint, prev_days, len_train = -1, days_ahead = 7):
    # The new version uses the repeat vector layer, and uses the
    # Time Distributed Wrapper at the end insted of at the beginning.
    # Thus, the model traines on a seven day output?

    '''
    :param ts: Input time series to be forecasted
    :param endpoint: Last index used for training the model
    :param prev_days: Amount of days used to predict the next days
    :param len_train: Amount of days back from endpoint used to train the model
    :param days_ahead: Amount of days forecasted after endpoint
    :return: days_ahead-forecast
    '''
    set_all_seeds(seed_value)
    if len_train == -1:
        len_train = endpoint
    scaler = MinMaxScaler()
    x_train = ts[endpoint - len_train:endpoint].reshape(len_train, 1)  # 17th July
    scaler.fit(x_train)
    x_train = scaler.transform(x_train)
    X, y = split_sequence(x_train, prev_days)
    # reshape from [samples, timesteps] into [samples, subsequences, timesteps, ...
        features]
    X = X.reshape((X.shape[0], prev_days, 1))

    preds = np.zeros((10, days_ahead))
    for i in range(10):
        start_time = time.time()
        model, loss = train_model(X, y, days_ahead, prev_days=prev_days)
        print("Time used to fit model", i+1, ": ", (time.time() - start_time), " ...
            sec")
        start_time = time.time()
        preds[i] = np.round_(k_step_prediction(
            days_ahead, x_train, model, scaler, prev_days=prev_days))
        print("Time used to forecast with model", i + 1, ": ",
              (time.time() - start_time), " sec")

    np.savetxt(os.getcwd() + r"\Predictions/nor_preds_" + str(days_ahead) +
```

```
               "_days_ahead_from_" + str(endpoint) + "_with_" + str(len_train) +
               "_train_data.csv", preds)
    return preds


def single_pred_with_test_data(days_ahead, x_train_and_test, model, scaler, ...
    prev_days = 7):
    '''
    Performes a series of one step predictions, one step at a time to generate a ...
        k-step prediction.
    At each iteration, the newly observed day is incorporated into the input of ...
        the prediction.
    :return: k-step prediction of covid-19 new cases
    '''
    predictions = []
    length_relevant_data = prev_days + days_ahead -1 # how far back to start the ...
        input

    # the first input to the prediction is the last prev_days elements of the ...
        train data
    for i in range(0, days_ahead-1):
        x_input = x_train_and_test[-length_relevant_data + ...
            i:-length_relevant_data + prev_days + i]
        x_input = x_input.reshape((1, prev_days, 1))
        yhat = model.predict(x_input, verbose=0)[0][0].reshape(1, 1)
        predictions.append(yhat)

    x_input = x_train_and_test[-prev_days:]
    x_input = x_input.reshape((1, prev_days, 1))
    yhat = model.predict(x_input, verbose=0)[0][0].reshape(1, 1)
    predictions.append(yhat)

    for i in range(len(predictions)):
        predictions[i] = scaler.inverse_transform(predictions[i])[0][0]
    return predictions

def one_step_preds_global(ts, endpoint, prev_days, len_train = -1, days_ahead = 7):
    set_all_seeds(seed_value)
    if len_train == -1:
        len_train = endpoint
    scaler = MinMaxScaler()
    x_train = ts[endpoint - len_train:endpoint].reshape(len_train, 1) # last ...
        len_train days used to predict
    scaler.fit(x_train)
    x_train = scaler.transform(x_train)
    # test data on day d is used to predict new cases on day d+1
    x_train_and_test = ts[endpoint - ...
        len_train:(endpoint+days_ahead-1)].reshape(len_train+days_ahead-1, 1)
    x_train_and_test = scaler.transform(x_train_and_test)
    # split train data into input-output matrix
    X, y = split_sequence(x_train, prev_days)
    X = X.reshape((X.shape[0], prev_days, 1))

    # Make 10 predictions and take mean forecast, to reduce variability in forecast
    preds = np.zeros((10, days_ahead))

    for i in range(10):
        start_time = time.time()
        model, loss = train_model(X, y, days_ahead, prev_days=prev_days) # The ...
            model is only trained in the train data!
        print("Time used to fit model", i+1, ": ", (time.time() - start_time), " ...
            sec")
        start_time = time.time()
        preds[i] = np.round_(single_pred_with_test_data(days_ahead, ...
            x_train_and_test, model, scaler, prev_days=prev_days))
        print("Time used to forecast with model", i+1, ": ", (time.time() - ...
            start_time), " sec")

    np.savetxt(os.getcwd() + r"\Predictions/glob_test_preds_" + str(days_ahead) +
               "_days_ahead_from_" + str(endpoint) + "_with_" + str(len_train) + ...
                   "_train_data.csv", preds)
    return preds

```

```
def one_step_preds_norway(ts, endpoint, prev_days, len_train = -1, days_ahead = 7):
    set_all_seeds(seed_value)
    if len_train == -1:
        len_train = endpoint
    scaler = MinMaxScaler()
    x_train = ts[endpoint - len_train:endpoint].reshape(len_train, 1) # last ...
        len_train days used to predict
    scaler.fit(x_train)
    x_train = scaler.transform(x_train)
    # test data on day d is used to predict new cases on day d+1
    x_train_and_test = ts[endpoint - ...
        len_train:(endpoint+days_ahead-1)].reshape(len_train+days_ahead-1, 1)
    x_train_and_test = scaler.transform(x_train_and_test)
    # split train data into input-output matrix
    X, y = split_sequence(x_train, prev_days)
    X = X.reshape((X.shape[0], prev_days, 1))

    # Make 10 predictions and take mean forecast, to reduce variability in forecast
    preds = np.zeros((10, days_ahead))
    for i in range(10):
        start_time = time.time()
        model, loss = train_model(X, y, days_ahead, prev_days=prev_days) # The ...
            model is only trained in the train data!
        print("Time used to fit model", i+1, ": ", (time.time() - start_time), " ...
            sec")
        start_time = time.time()
        preds[i] = np.round_(single_pred_with_test_data(days_ahead, ...
            x_train_and_test, model, scaler, prev_days=prev_days))
        print("Time used to forecast with model", i+1, ": ", (time.time() - ...
            start_time), " sec")

    np.savetxt(os.getcwd() + r"\Predictions/nor_test_preds_" + str(days_ahead) +
               "_days_ahead_from_" + str(endpoint) + "_with_" + str(len_train) + ...
                   "_train_data.csv", preds)
    return preds

def one_step_preds_on_train_global(ts, endpoint, prev_days, len_train = -1, ...
    days_ahead = 7):
    set_all_seeds(seed_value)
    if len_train == -1:
        len_train = endpoint
    scaler = MinMaxScaler()
    x_train = ts[endpoint - len_train:endpoint].reshape(len_train, 1) # last ...
        len_train days used to predict
    scaler.fit(x_train)
    x_train = scaler.transform(x_train)
    # test data on day d is used to predict new cases on day d+1
    x_train_and_test = ts[endpoint - ...
        len_train:(endpoint+days_ahead-1)].reshape(len_train+days_ahead-1, 1)
    x_train_and_test = scaler.transform(x_train_and_test)
    # split train data into input-output matrix
    X, y = split_sequence(x_train, prev_days)
    X = X.reshape((X.shape[0], prev_days, 1))
    # Make 10 predictions and take mean forecast, to reduce variability in forecast
    preds = np.zeros((10, days_ahead))
    for i in range(10):
        start_time = time.time()
        model, loss = train_model(X, y, days_ahead, prev_days=prev_days) # The ...
            model is only trained in the train data!
        print("Time used to fit model", i+1, ": ", (time.time() - start_time), " ...
            sec")
        start_time = time.time()
        preds[i] = np.round_(single_pred_with_test_data(days_ahead, ...
            x_train_and_test[:-days_ahead], model, scaler, prev_days=prev_days))
        print("Time used to forecast with model", i+1, ": ", (time.time() - ...
            start_time), " sec")

    np.savetxt(os.getcwd() + r"\Predictions/glob_test_preds_train_" + ...
        str(days_ahead) +
               "_from_" + str(endpoint) + "_with_" + str(len_train) + ...
                   "_train_data.csv", preds)
    return preds

```

```
def one_step_preds_on_train_norway(ts, endpoint, prev_days, len_train = -1, ...
    days_ahead = 7):
    set_all_seeds(seed_value)
    if len_train == -1:
        len_train = endpoint
    scaler = MinMaxScaler()
    x_train = ts[endpoint - len_train:endpoint].reshape(len_train, 1) # last ...
        len_train days used to predict
    scaler.fit(x_train)
    x_train = scaler.transform(x_train)
    # test data on day d is used to predict new cases on day d+1
    x_train_and_test = ts[endpoint - ...
        len_train:(endpoint+days_ahead-1)].reshape(len_train+days_ahead-1, 1)
    x_train_and_test = scaler.transform(x_train_and_test)
    # split train data into input-output matrix
    X, y = split_sequence(x_train, prev_days)
    X = X.reshape((X.shape[0], prev_days, 1))
    # Make 10 predictions and take mean forecast, to reduce variability in forecast
    preds = np.zeros((10, days_ahead))
    for i in range(10):
        start_time = time.time()
        model, loss = train_model(X, y, days_ahead, prev_days=prev_days) # The ...
            model is only trained in the train data!
        print("Time used to fit model", i+1, ": ", (time.time() - start_time), " ...
            sec")
        start_time = time.time()
        preds[i] = np.round_(single_pred_with_test_data(days_ahead, ...
            x_train_and_test[:-days_ahead], model, scaler, prev_days=prev_days))
        print("Time used to forecast with model", i+1, ": ", (time.time() - ...
            start_time), " sec")

    np.savetxt(os.getcwd() + r"\Predictions/nor_test_preds_on_train_" + ...
        str(days_ahead) +
               "_from_" + str(endpoint) + "_with_" + str(len_train) + ...
                   "_train_data.csv", preds)
    return preds

def all_in_one_global(endpoint, days_ahead, len_train):
    # Wrapper function to systemize predictions results
    # The function will output both the one-step prediction and the days_ahead ...
        prediction
    prediction_global(ts=ts_global, endpoint=endpoint, prev_days=7, ...
        days_ahead=days_ahead, len_train=len_train)
    one_step_preds_global(ts=ts_global, endpoint=endpoint, prev_days=7, ...
        days_ahead = days_ahead, len_train = len_train)


def all_in_one_norway(endpoint, days_ahead, len_train):
    # Wrapper function to systemize predictions,
    # The function will output both the one-step prediction and the days_ahead ...
        prediction
    one_step_preds_norway(ts=ts_norway, endpoint=endpoint, prev_days=7, ...
        days_ahead = days_ahead, len_train = len_train)
    prediction_norway(ts=ts_norway, endpoint=endpoint, prev_days=7, days_ahead = ...
        days_ahead, len_train = len_train)

def accuracy_with_simulations_global(ts, simulations, endpoint, days_ahead, ...
    prev_days = 7, len_train = -1):
    '''
    Forecasts with scheme 1, but use simulations instead of observations
    :param ts: The time series of interest
    :param simulations: Simulated observations from Gandalf model, imported from ...
        Matlab
    :param endpoint: Last index of dataset used in training
    :param prev_days: how many days put in to the model (7 is default)
    :param days_ahead: days ahead forecast
    :return: the RRMSEs and MAPEs associated with each simulation
    '''
    set_all_seeds(seed_value)

    simulations = np.array(simulations)
    if len_train == -1:
        len_train = endpoint
```

```
    scaler = MinMaxScaler()
    x_train = ts[endpoint - len_train:endpoint].reshape(len_train, 1)  # last ...
        len_train days used to predict
    scaler.fit(x_train)
    x_train = scaler.transform(x_train)
    # split train data into input-output matrix
    X, y = split_sequence(x_train, prev_days)
    X = X.reshape((X.shape[0], prev_days, 1))

    # Make 10 predictions and take mean forecast, to reduce variability in forecast
    RRMSEs = np.zeros(simulations.shape[0])
    MAPEs = np.zeros(simulations.shape[0])

    models = [] # This saves a lot of time!
    for j in range(10):
        model, loss = train_model(X, y, days_ahead,
                                  prev_days=prev_days)  # The model is only ...
                                      trained in the train data!
        models.append(model)

    for i in range(simulations.shape[0]):
        start_time = time.time()
        simulation = scaler.transform(simulations[i].reshape(-1, 1))
        input_in_prediction = np.concatenate((x_train, simulation[0:-1]), axis=0)
        ten_preds = np.zeros((10, days_ahead))
        for j in range(10):
            ten_preds[j] = np.round_(
                single_pred_with_test_data(days_ahead, input_in_prediction, ...
                    models[j], scaler, prev_days=prev_days))
        mean_pred = np.mean(ten_preds, axis = 0)

        simulation = scaler.inverse_transform(simulation.reshape(-1, 1))
        mean_sim = simulation.mean()
        MSE = np.square(np.subtract(simulation, mean_pred)).mean()
        RRMSEs[i] = np.round_(np.sqrt(MSE) / mean_sim * 100, decimals=3)
        mape = tf.keras.losses.MeanAbsolutePercentageError()
        MAPEs[i] = np.round_(mape(simulation, mean_pred).numpy(), decimals=3)
        print("Time used to predict with simulation", i, ": ", (time.time() - ...
            start_time), " sec")

    np.savetxt(os.getcwd() + r"\Predictions/RRMSEs_from_simulation_global_" + ...
        str(days_ahead) +
               "_days_ahead_with_" + str(len_train) + "_train_data_from" + ...
                   str(endpoint) + ".csv", RRMSEs)
    np.savetxt(os.getcwd() + r"\Predictions/MAPEs_from_simulation_global_" + ...
        str(days_ahead) +
               "_days_ahead_with_" + str(len_train) + "_train_data_from" + ...
                   str(endpoint) + ".csv", MAPEs)


def predictions_with_simulations_global(ts, simulations, endpoint, days_ahead, ...
    prev_days = 7, len_train = -1):
    '''
    Forecasts with scheme 1, but use simulations instead of observations, the ...
        accuracy measures are returned
    :param ts: The time series of interest
    :param simulations: Simulated observations from Gandalf model, imported from ...
        Matlab
    :param endpoint: Last index of dataset used in training
    :param prev_days: how many days put in to the model (7 is default)
    :param days_ahead: days ahead forecast
    :return: the RRMSEs and MAPEs associated with each simulation
    '''
    set_all_seeds(seed_value)

    simulations = np.array(simulations)
    if len_train == -1:
        len_train = endpoint
    scaler = MinMaxScaler()
    x_train = ts[endpoint - len_train:endpoint].reshape(len_train, 1)  # last ...
        len_train days used to predict
    scaler.fit(x_train)
    x_train = scaler.transform(x_train)
```

```
    # split train data into input-output matrix
    X, y = split_sequence(x_train, prev_days)
    X = X.reshape((X.shape[0], prev_days, 1))

    # Make 10 predictions and take mean forecast, to reduce variability in forecast
    preds_matlab = np.zeros((3, 28))

    start_time = time.time()

    models = []  # This saves a lot of time!
    for j in range(10):
        model, loss = train_model(X, y, days_ahead,
                                  prev_days=prev_days)  # The model is only ...
                                      trained in the train data!
        models.append(model)
    print("time to fit ten models: ", time.time() - start_time)

    for i in range(3):  # for the first three predictions
        preds = np.zeros((10, 28))
        print("simulations[i]", simulations[i])
        simulation = scaler.transform(simulations[i].reshape(-1, 1))
        input_in_prediction = np.concatenate((x_train, simulation[0:-1]), axis=0)
        for j in range(10):
            pred = np.round_(
                single_pred_with_test_data(days_ahead, input_in_prediction, ...
                    models[j], scaler, prev_days=prev_days))
            preds[j] = pred
        preds_matlab[i] = np.mean(preds, axis = 0)
    print("Time used to predict all simulations: ", (time.time() - start_time), " ...
        sec")

    print(preds_matlab, preds_matlab.shape)
    np.savetxt(os.getcwd() + r"\Predictions/pred_three_preds_on_sims.csv", ...
        preds_matlab)


def predict_weekday_no_test(Xs, Ys, x_trains, days_ahead, scaler, prev_days = 7):
    '''
    The function predicts a days_ahead forecast for seven models, one for each ...
        weekday.
    :param datas: list of seven scaled train data, one containing all Mondays etc...
    :param days_ahead: how many days ahead to predict
    :param scaler: pre-trained scaler function
    :param prev_days: how many days used to predict preceding day for each submodel
    :return: days_ahead forecast
    '''

    prediction = np.zeros(days_ahead)
    prediction_days = [i for i in range(days_ahead)] # sequence of numbers
    for i in range(len(Xs)):
        days_to_predict = prediction_days[i::7] # divides what days to predict by ...
            each submodel, the length is the amount of days to predict
        X = Xs[i]
        y = Ys[i]
        x_train = x_trains[i]
        model, loss = train_model(X, y, len(days_to_predict), prev_days)
        prediction[days_to_predict] = k_step_prediction(len(days_to_predict), ...
            x_train, model, scaler, prev_days) # prediction of all of a single ...
            weekday
    return prediction




def weekday_models_no_test_data_global(ts, endpoint, days_ahead, prev_days = 7, ...
    len_train = -1):
    set_all_seeds(seed_value)
    if len_train == -1:
        len_train = endpoint
    scaler = MinMaxScaler()
    x_train = ts[endpoint - len_train:endpoint].reshape(len_train, 1)  # use ...
        train data to fit scaler
    scaler.fit(x_train)
```

```
    x_train = scaler.transform(x_train)
    x_train_1 = x_train[::7] # Every 7th day
    x_train_2 = x_train[1::7]
    x_train_3 = x_train[2::7]
    x_train_4 = x_train[3::7]
    x_train_5 = x_train[4::7]
    x_train_6 = x_train[5::7]
    x_train_7 = x_train[6::7]
    print(x_train_1.shape, x_train_2.shape, x_train_3.shape, x_train_4.shape, ...
        x_train_5.shape, x_train_6.shape,x_train_7.shape)

    x_trains = [x_train_1, x_train_2, x_train_3, x_train_4, x_train_5, x_train_6, ...
        x_train_7]
    X_1, y_1 = split_sequence(x_train_1, prev_days)
    X_2, y_2 = split_sequence(x_train_2, prev_days)
    X_3, y_3 = split_sequence(x_train_3, prev_days)
    X_4, y_4 = split_sequence(x_train_4, prev_days)
    X_5, y_5 = split_sequence(x_train_5, prev_days)
    X_6, y_6 = split_sequence(x_train_6, prev_days)
    X_7, y_7 = split_sequence(x_train_7, prev_days)

    Ys = [y_1, y_2, y_3, y_4, y_5, y_6, y_7]
    # reshape from [samples, timesteps] into [samples, subsequences, timesteps, ...
        features]
    X_1 = X_1.reshape((X_1.shape[0], prev_days, 1))
    X_2 = X_2.reshape((X_2.shape[0], prev_days, 1))
    X_3 = X_3.reshape((X_3.shape[0], prev_days, 1))
    X_4 = X_4.reshape((X_4.shape[0], prev_days, 1))
    X_5 = X_5.reshape((X_5.shape[0], prev_days, 1))
    X_6 = X_6.reshape((X_6.shape[0], prev_days, 1))
    X_7 = X_7.reshape((X_7.shape[0], prev_days, 1))
    Xs =[X_1, X_2, X_3, X_4, X_5, X_6, X_7]
    preds = np.zeros((10, days_ahead))

    for i in range(10):
        start_time = time.time()
        preds[i] = np.round_(predict_seven_no_test(Xs, Ys, x_trains, days_ahead, ...
            scaler, prev_days))
        print("Time used to fit and forecast with model", i + 1, ": ", ...
            (time.time() - start_time), " sec")

    np.savetxt(os.getcwd() + r"\Predictions/glob_weekday_preds_" + ...
        str(days_ahead) +
               "_days_ahead_from_" + str(endpoint) + "_with_" + str(len_train) + ...
                   "_train_data.csv", preds)

def predict_weekday_with_test(Xs, Ys, x_train_and_tests, days_ahead, scaler, ...
    prev_days = 7):
    '''
    The function predicts a days_ahead forecast for seven models, one for each ...
        weekday.
    :param datas: list of seven scaled train data, one containing all Mondays etc...
    :param days_ahead: how many days ahead to predict
    :param scaler: pre-trained scaler function
    :param prev_days: how many days used to predict preceding day for each submodel
    :return: days_ahead forecast
    '''

    prediction = np.zeros(days_ahead)
    prediction_days = [i for i in range(days_ahead)] # sequence of numbers
    for i in range(len(Xs)):
        days_to_predict = prediction_days[i::7] # divides what days to predict by ...
            each submodel, the length is the amount of days to predict
        X = Xs[i]
        y = Ys[i]
        x_train_and_test = x_train_and_tests[i]
        model, loss = train_model(X, y, len(days_to_predict), prev_days)
        # prediction of all of a single weekday:
        prediction[days_to_predict] = \
            single_pred_with_test_data(len(days_to_predict), x_train_and_test, ...
                model, scaler, prev_days)
    return prediction

```

```
def weekday_models_with_test_data_global(ts, endpoint, days_ahead, prev_days = 7, ...
    len_train = -1):
    set_all_seeds(seed_value)
    if len_train == -1:
        len_train = endpoint
    scaler = MinMaxScaler()
    x_train = ts[endpoint - len_train:endpoint].reshape(len_train, 1)  # use ...
        train data to fit scaler
    scaler.fit(x_train)
    x_train = scaler.transform(x_train)
    x_train_1 = x_train[::7] # Every 7th day
    x_train_2 = x_train[1::7]
    x_train_3 = x_train[2::7]
    x_train_4 = x_train[3::7]
    x_train_5 = x_train[4::7]
    x_train_6 = x_train[5::7]
    x_train_7 = x_train[6::7]

    x_train_and_test = ts[endpoint - ...
        len_train:(endpoint+days_ahead-1)].reshape(len_train+days_ahead-1, 1)
    x_train_and_test = scaler.transform(x_train_and_test)
    x_train_and_test_1 = x_train_and_test[::7] # Every 7th day
    x_train_and_test_2 = x_train_and_test[1::7]
    x_train_and_test_3 = x_train_and_test[2::7]
    x_train_and_test_4 = x_train_and_test[3::7]
    x_train_and_test_5 = x_train_and_test[4::7]
    x_train_and_test_6 = x_train_and_test[5::7]
    x_train_and_test_7 = x_train_and_test[6::7]

    x_train_and_tests = [x_train_and_test_1, x_train_and_test_2, ...
        x_train_and_test_3, x_train_and_test_4,
                x_train_and_test_5, x_train_and_test_6, x_train_and_test_7]

    X_1, y_1 = split_sequence(x_train_1, prev_days)
    X_2, y_2 = split_sequence(x_train_2, prev_days)
    X_3, y_3 = split_sequence(x_train_3, prev_days)
    X_4, y_4 = split_sequence(x_train_4, prev_days)
    X_5, y_5 = split_sequence(x_train_5, prev_days)
    X_6, y_6 = split_sequence(x_train_6, prev_days)
    X_7, y_7 = split_sequence(x_train_7, prev_days)

    Ys = [y_1, y_2, y_3, y_4, y_5, y_6, y_7]
    # reshape from [samples, timesteps] into [samples, subsequences, timesteps, ...
        features]
    X_1 = X_1.reshape((X_1.shape[0], prev_days, 1))
    X_2 = X_2.reshape((X_2.shape[0], prev_days, 1))
    X_3 = X_3.reshape((X_3.shape[0], prev_days, 1))
    X_4 = X_4.reshape((X_4.shape[0], prev_days, 1))
    X_5 = X_5.reshape((X_5.shape[0], prev_days, 1))
    X_6 = X_6.reshape((X_6.shape[0], prev_days, 1))
    X_7 = X_7.reshape((X_7.shape[0], prev_days, 1))
    Xs =[X_1, X_2, X_3, X_4, X_5, X_6, X_7]
    preds = np.zeros((10, days_ahead))

    for i in range(10):
        start_time = time.time()
        preds[i] = np.round_(predict_weekday_with_test(Xs, Ys, x_train_and_tests, ...
            days_ahead, scaler, prev_days))
        print("Time used to fit and forecast with model", i + 1, ": ", ...
            (time.time() - start_time), " sec")

    np.savetxt(os.getcwd() + r"\Predictions/glob_weekday_preds_test_" + ...
        str(days_ahead) +
               "_days_ahead_from_" + str(endpoint) + "_with_" + str(len_train) + ...
                   "_train_data.csv", preds)




############################### Global ##################################

# 28 day predictions global

```

```
###############
all_in_one_global( endpoint = 40, days_ahead = 28, len_train = 40)
# all_in_one_global( endpoint = 196, days_ahead = 28, len_train = 196)
# all_in_one_global( endpoint = 196, days_ahead = 28, len_train = 50)
# all_in_one_global( endpoint = 196, days_ahead = 28, len_train = 100)
#
# ##############
# all_in_one_global(endpoint = 100, days_ahead = 28, len_train = 100)
# all_in_one_global(endpoint = 300, days_ahead = 28, len_train = 300)
# all_in_one_global(endpoint = 500, days_ahead = 28, len_train = 500)
# all_in_one_global(endpoint = 743, days_ahead = 28, len_train = 743) # newest
#

# # Fewer observations: How robust is the model?
# all_in_one_global(endpoint = 100, days_ahead = 28, len_train = 50)
# all_in_one_global(endpoint = 300, days_ahead = 28, len_train = 50)
# all_in_one_global(endpoint = 300, days_ahead = 28, len_train = 100)
# all_in_one_global(endpoint = 500, days_ahead = 28, len_train = 50)
# all_in_one_global(endpoint = 500, days_ahead = 28, len_train = 100)
# all_in_one_global(endpoint = 743, days_ahead = 28, len_train = 50)
# all_in_one_global(endpoint = 743, days_ahead = 28, len_train = 100)

############################### Norway ###################################

# An equivalent set of predictions can be made on the norwegian data set.
# Start at 264 to compare with original Gandalf model.

# all_in_one_norway(endpoint = 264, days_ahead = 28, len_train = 264)
# all_in_one_norway(endpoint = 264, days_ahead = 28, len_train = 50)
# all_in_one_norway(endpoint = 264, days_ahead = 28, len_train = 100)
#
# all_in_one_norway(endpoint = 100, days_ahead = 28, len_train = 100)
# all_in_one_norway(endpoint = 300, days_ahead = 28, len_train = 300)
# all_in_one_norway(endpoint = 500, days_ahead = 28, len_train = 500)
# all_in_one_norway(endpoint = 696, days_ahead = 28, len_train = 696)
#
# # Fewer observations: How robust is the model?
# all_in_one_norway(endpoint = 100, days_ahead = 28, len_train = 50)
# all_in_one_norway(endpoint = 300, days_ahead = 28, len_train = 50)
# all_in_one_norway(endpoint = 300, days_ahead = 28, len_train = 100)
# all_in_one_norway(endpoint = 500, days_ahead = 28, len_train = 50)
# all_in_one_norway(endpoint = 500, days_ahead = 28, len_train = 100)
# all_in_one_norway(endpoint = 696, days_ahead = 28, len_train = 50)
# all_in_one_norway(endpoint = 696, days_ahead = 28, len_train = 100)


######################### Simulation study ############################

simulations = pd.read_csv(
    os.getcwd() + r"\Matlab\Saved variables\simulations_from_gandalf.csv", ...
        header=None)


# print("simulation.shape = ", simulations.shape)
#print("simulations.iloc[0:2, :] = ", simulations.iloc[0:3, :])
# accuracy_with_simulations_global(ts_global, simulations, endpoint=196, ...
    days_ahead=28)
# predictions_with_simulations_global(ts_global, simulations.iloc[0:3, ...
    :], endpoint=196, days_ahead=28)
# predictions_with_simulations_global(ts_norway, simulations.iloc[0:3, ...
    :], endpoint=196, days_ahead=28)

######################## Predict with weekday model  #####################

# weekday_models_no_test_data_global(ts = ts_global, endpoint = 196, days_ahead = 7)
# weekday_models_with_test_data_global(ts = ts_global, endpoint = 600, days_ahead ...
    = 7)
```

Matlab-code

The following code was used to derive and present all the results of the thesis, except for the predictions from the CNN-LSTM model.

```
warning('off','all')
%%%%%%%%%%%%%%% Import data sets %%%%%%%%%%%%%%%

Norwegian_data_set = table2array(readtable("Data\new_cases_Norway.csv"));
global_data_set = table2array(readtable("Data\new_cases_global.csv"));

% Transform to log scale imediately
ts = log(max(0.1, Norwegian_data_set));
ts_global = log(max(0.1, global_data_set));

%%%%% General dictionary: %%%%%

% endpoint: last index included as training data

% len_train: amount of training data

% days_ahead: length of forecast ensuing endpoint

% With_observed: predictioon scheme 1

% no_observed: prediction scheme 2

%%%%%%%%%%%%%%%%%%%%%%%




% Collecting predictions from CNN-LSTM model from Python

%%%%%%%%%%%%%%%%%%%% Gather Global Forecasts %%%%%%%%%%%%%%%%%%%

loss_func = table2array(readtable(...
    "Predictions\mean_loss_on_original_test_data.csv"));

% global 28 day forecast with prediction scheme 1
glob_test_preds_28_days_ahead_from_100_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\glob_test_preds_28_days_ahead_from_100_with_50_train_data.csv'));
glob_test_preds_28_days_ahead_from_100_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\glob_test_preds_28_days_ahead_from_100_with_100_train_data.csv'));

glob_test_preds_28_days_ahead_from_196_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\glob_test_preds_28_days_ahead_from_196_with_50_train_data.csv'));
glob_test_preds_28_days_ahead_from_196_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\glob_test_preds_28_days_ahead_from_196_with_100_train_data.csv'));
glob_test_preds_28_days_ahead_from_196_with_196_train_data = ...
    table2array(readtable(...
    'Predictions\glob_test_preds_28_days_ahead_from_196_with_196_train_data.csv'));

glob_test_preds_28_days_ahead_from_300_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\glob_test_preds_28_days_ahead_from_300_with_50_train_data.csv'));
glob_test_preds_28_days_ahead_from_300_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\glob_test_preds_28_days_ahead_from_300_with_100_train_data.csv'));
glob_test_preds_28_days_ahead_from_300_with_300_train_data = ...
    table2array(readtable(...
    'Predictions\glob_test_preds_28_days_ahead_from_300_with_300_train_data.csv'));

glob_test_preds_28_days_ahead_from_500_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\glob_test_preds_28_days_ahead_from_500_with_50_train_data.csv'));
glob_test_preds_28_days_ahead_from_500_with_100_train_data = ...
```

```
    table2array(readtable(...
    'Predictions\glob_test_preds_28_days_ahead_from_500_with_100_train_data.csv'));
glob_test_preds_28_days_ahead_from_500_with_500_train_data = ...
    table2array(readtable(...
    'Predictions\glob_test_preds_28_days_ahead_from_500_with_500_train_data.csv'));

glob_test_preds_28_days_ahead_from_743_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\glob_test_preds_28_days_ahead_from_500_with_500_train_data.csv'));
glob_test_preds_28_days_ahead_from_743_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\glob_test_preds_28_days_ahead_from_743_with_100_train_data.csv'));
glob_test_preds_28_days_ahead_from_743_with_743_train_data = ...
    table2array(readtable(...
    'Predictions\glob_test_preds_28_days_ahead_from_743_with_743_train_data.csv'));


% global seven and 28 day forecasts with prediction scheme 2
glob_preds_28_days_ahead_from_100_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\glob_preds_28_days_ahead_from_100_with_50_train_data.csv'));
glob_preds_7_days_ahead_from_100_with_50_train_data = ...
    glob_preds_28_days_ahead_from_100_with_50_train_data(:, 1:7);
glob_preds_28_days_ahead_from_100_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\glob_preds_28_days_ahead_from_100_with_100_train_data.csv'));
glob_preds_7_days_ahead_from_100_with_100_train_data = ...
    glob_preds_28_days_ahead_from_100_with_100_train_data(:, 1:7);

glob_preds_28_days_ahead_from_196_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\glob_preds_28_days_ahead_from_196_with_50_train_data.csv'));
glob_preds_7_days_ahead_from_196_with_50_train_data = ...
    glob_preds_28_days_ahead_from_196_with_50_train_data(:, 1:7);
glob_preds_28_days_ahead_from_196_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\glob_preds_28_days_ahead_from_196_with_100_train_data.csv'));
glob_preds_7_days_ahead_from_196_with_100_train_data = ...
    glob_preds_28_days_ahead_from_196_with_100_train_data(:, 1:7);
glob_preds_28_days_ahead_from_196_with_196_train_data = ...
    table2array(readtable(...
    'Predictions\glob_preds_28_days_ahead_from_196_with_196_train_data.csv'));
glob_preds_7_days_ahead_from_196_with_196_train_data = ...
    glob_preds_28_days_ahead_from_196_with_196_train_data(:, 1:7);

glob_preds_28_days_ahead_from_300_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\glob_preds_28_days_ahead_from_300_with_50_train_data.csv'));
glob_preds_7_days_ahead_from_300_with_50_train_data = ...
    glob_preds_28_days_ahead_from_300_with_50_train_data(:, 1:7);
glob_preds_28_days_ahead_from_300_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\glob_preds_28_days_ahead_from_300_with_100_train_data.csv'));
glob_preds_7_days_ahead_from_300_with_100_train_data = ...
    glob_preds_28_days_ahead_from_300_with_100_train_data(:, 1:7);
glob_preds_28_days_ahead_from_300_with_300_train_data = ...
    table2array(readtable(...
    'Predictions\glob_preds_28_days_ahead_from_300_with_300_train_data.csv'));
glob_preds_7_days_ahead_from_300_with_300_train_data = ...
    glob_preds_28_days_ahead_from_300_with_300_train_data(:, 1:7);

glob_preds_28_days_ahead_from_500_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\glob_preds_28_days_ahead_from_500_with_50_train_data.csv'));
glob_preds_7_days_ahead_from_500_with_50_train_data = ...
    glob_preds_28_days_ahead_from_500_with_50_train_data(:, 1:7);
glob_preds_28_days_ahead_from_500_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\glob_preds_28_days_ahead_from_500_with_100_train_data.csv'));
glob_preds_7_days_ahead_from_500_with_100_train_data = ...
    glob_preds_28_days_ahead_from_500_with_100_train_data(:, 1:7);
glob_preds_28_days_ahead_from_500_with_500_train_data = ...
    table2array(readtable(...
```

```
    'Predictions\glob_preds_28_days_ahead_from_500_with_500_train_data.csv'));
glob_preds_7_days_ahead_from_500_with_500_train_data = ...
    glob_preds_28_days_ahead_from_500_with_500_train_data(:, 1:7);

glob_preds_28_days_ahead_from_743_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\glob_preds_28_days_ahead_from_743_with_50_train_data.csv'));
glob_preds_7_days_ahead_from_743_with_50_train_data = ...
    glob_preds_28_days_ahead_from_743_with_50_train_data(:, 1:7);
glob_preds_28_days_ahead_from_743_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\glob_preds_28_days_ahead_from_743_with_100_train_data.csv'));
glob_preds_7_days_ahead_from_743_with_100_train_data = ...
    glob_preds_28_days_ahead_from_743_with_100_train_data(:, 1:7);
glob_preds_28_days_ahead_from_743_with_743_train_data = ...
    table2array(readtable(...
    'Predictions\glob_preds_28_days_ahead_from_743_with_743_train_data.csv'));
glob_preds_7_days_ahead_from_743_with_743_train_data = ...
    glob_preds_28_days_ahead_from_743_with_743_train_data(:, 1:7);


%%%%%%%%%%%%%%%%%%%% Gather Norwegian Forecasts %%%%%%%%%%%%%%%%%%%%

% Norway 28 day forecast with prediction scheme 1
nor_test_preds_28_days_ahead_from_100_with_50_train_data =  ...
    table2array(readtable(...
    'Predictions\nor_test_preds_28_days_ahead_from_100_with_50_train_data.csv'));
nor_test_preds_28_days_ahead_from_100_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\nor_test_preds_28_days_ahead_from_100_with_100_train_data.csv'));

nor_test_preds_28_days_ahead_from_264_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\nor_test_preds_28_days_ahead_from_264_with_50_train_data.csv'));
nor_test_preds_28_days_ahead_from_264_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\nor_test_preds_28_days_ahead_from_264_with_100_train_data.csv'));
nor_test_preds_28_days_ahead_from_264_with_264_train_data = ...
    table2array(readtable(...
    'Predictions\nor_test_preds_28_days_ahead_from_264_with_264_train_data.csv'));

nor_test_preds_28_days_ahead_from_300_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\nor_test_preds_28_days_ahead_from_300_with_50_train_data.csv'));
nor_test_preds_28_days_ahead_from_300_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\nor_test_preds_28_days_ahead_from_300_with_100_train_data.csv'));
nor_test_preds_28_days_ahead_from_300_with_300_train_data = ...
    table2array(readtable(...
    'Predictions\nor_test_preds_28_days_ahead_from_300_with_300_train_data.csv'));

nor_test_preds_28_days_ahead_from_500_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\nor_test_preds_28_days_ahead_from_500_with_50_train_data.csv'));
nor_test_preds_28_days_ahead_from_500_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\nor_test_preds_28_days_ahead_from_500_with_100_train_data.csv'));
nor_test_preds_28_days_ahead_from_500_with_500_train_data = ...
    table2array(readtable(...
    'Predictions\nor_test_preds_28_days_ahead_from_500_with_500_train_data.csv'));

nor_test_preds_28_days_ahead_from_696_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\nor_test_preds_28_days_ahead_from_696_with_50_train_data.csv'));
nor_test_preds_28_days_ahead_from_696_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\nor_test_preds_28_days_ahead_from_696_with_100_train_data.csv'));
nor_test_preds_28_days_ahead_from_696_with_696_train_data = ...
    table2array(readtable(...
    'Predictions\nor_test_preds_28_days_ahead_from_696_with_696_train_data.csv'));


% Norway seven (and 28) day forecasts with prediction scheme 2
```

```
nor_preds_28_days_ahead_from_100_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\nor_preds_28_days_ahead_from_100_with_50_train_data.csv'));
nor_preds_7_days_ahead_from_100_with_50_train_data = ...
    nor_preds_28_days_ahead_from_100_with_50_train_data(:, 1:7);
nor_preds_28_days_ahead_from_100_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\nor_preds_28_days_ahead_from_100_with_100_train_data.csv'));
nor_preds_7_days_ahead_from_100_with_100_train_data = ...
    nor_preds_28_days_ahead_from_100_with_100_train_data(:, 1:7);

nor_preds_28_days_ahead_from_264_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\nor_preds_28_days_ahead_from_264_with_50_train_data.csv'));
nor_preds_7_days_ahead_from_264_with_50_train_data = ...
    nor_preds_28_days_ahead_from_264_with_50_train_data(:, 1:7);
nor_preds_28_days_ahead_from_264_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\nor_preds_28_days_ahead_from_264_with_100_train_data.csv'));
nor_preds_7_days_ahead_from_264_with_100_train_data = ...
    nor_preds_28_days_ahead_from_264_with_100_train_data(:, 1:7);
nor_preds_28_days_ahead_from_264_with_264_train_data = ...
    table2array(readtable(...
    'Predictions\nor_preds_28_days_ahead_from_264_with_264_train_data.csv'));
nor_preds_7_days_ahead_from_264_with_264_train_data = ...
    nor_preds_28_days_ahead_from_264_with_264_train_data(:, 1:7);

nor_preds_28_days_ahead_from_300_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\nor_preds_28_days_ahead_from_300_with_50_train_data.csv'));
nor_preds_7_days_ahead_from_300_with_50_train_data = ...
    nor_preds_28_days_ahead_from_300_with_50_train_data(:, 1:7);
nor_preds_28_days_ahead_from_300_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\nor_preds_28_days_ahead_from_300_with_100_train_data.csv'));
nor_preds_7_days_ahead_from_300_with_100_train_data = ...
    nor_preds_28_days_ahead_from_300_with_100_train_data(:, 1:7);
nor_preds_28_days_ahead_from_300_with_300_train_data = ...
    table2array(readtable(...
    'Predictions\nor_preds_28_days_ahead_from_300_with_300_train_data.csv'));
nor_preds_7_days_ahead_from_300_with_300_train_data = ...
    nor_preds_28_days_ahead_from_300_with_300_train_data(:, 1:7);

nor_preds_28_days_ahead_from_500_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\nor_preds_28_days_ahead_from_500_with_50_train_data.csv'));
nor_preds_7_days_ahead_from_500_with_50_train_data = ...
    nor_preds_28_days_ahead_from_500_with_50_train_data(:, 1:7);
nor_preds_28_days_ahead_from_500_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\nor_preds_28_days_ahead_from_500_with_100_train_data.csv'));
nor_preds_7_days_ahead_from_500_with_100_train_data = ...
    nor_preds_28_days_ahead_from_500_with_100_train_data(:, 1:7);
nor_preds_28_days_ahead_from_500_with_500_train_data = ...
    table2array(readtable(...
    'Predictions\nor_preds_28_days_ahead_from_500_with_500_train_data.csv'));
nor_preds_7_days_ahead_from_500_with_500_train_data = ...
    nor_preds_28_days_ahead_from_500_with_500_train_data(:, 1:7);

nor_preds_28_days_ahead_from_696_with_50_train_data = ...
    table2array(readtable(...
    'Predictions\nor_preds_28_days_ahead_from_696_with_50_train_data.csv'));
nor_preds_7_days_ahead_from_696_with_50_train_data = ...
    nor_preds_28_days_ahead_from_696_with_50_train_data(:, 1:7);
nor_preds_28_days_ahead_from_696_with_100_train_data = ...
    table2array(readtable(...
    'Predictions\nor_preds_28_days_ahead_from_696_with_100_train_data.csv'));
nor_preds_7_days_ahead_from_696_with_100_train_data = ...
    nor_preds_28_days_ahead_from_696_with_100_train_data(:, 1:7);
nor_preds_28_days_ahead_from_696_with_696_train_data = ...
    table2array(readtable(...
    'Predictions\nor_preds_28_days_ahead_from_696_with_696_train_data.csv'));
nor_preds_7_days_ahead_from_696_with_696_train_data = ...
```

```
    nor_preds_28_days_ahead_from_696_with_696_train_data(:, 1:7);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Plots %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% plot_norway_and_global_time_series_and_transform(ts, ts_global, false)
% %
% plot_norway_and_global_time_series_and_transform(ts, ts_global, true)


%%%%%%%%%%%%%%%%%%%% Norwegian Forecasts %%%%%%%%%%%%%%%%%%%%

% plot_preds_sarima_gandalf(ts, 264, 264, 7, false)
% plot_acf_of_res(ts, 264, 264, false)
% plot_acf_of_res(ts, 264, 264, true)

% plot_10_norway_CNN_LSTM_Preds(floor(exp(ts)), ...
%     nor_preds_7_days_ahead_from_264_with_264_train_data, 264, 264, 7, false)
% plot_preds_without_observed(ts, ...
%     mean(nor_preds_7_days_ahead_from_264_with_264_train_data, 1), ...
%     264, 264, 7, false, 14)
% plot_10_norway_CNN_LSTM_Preds(floor(exp(ts)), ...
%     nor_test_preds_28_days_ahead_from_264_with_264_train_data, ...
%     264, 264, 28, false)
% plot_preds_with_observed(ts, ...
%     mean(nor_test_preds_28_days_ahead_from_264_with_264_train_data, 1), ...
%     264, 264, 28, false, 30)
%
%
% plot_parameters_from_prediction_scheme_1(ts, 264, 264, 28, false)
%
% plot_preds_without_observed(ts, ...
%     mean(nor_preds_7_days_ahead_from_264_with_50_train_data, 1), ...
%     264, 50, 7, false, 14)
% plot_preds_with_observed(ts, ...
%     mean(nor_test_preds_28_days_ahead_from_264_with_50_train_data, 1), ...
%     264, 50, 28, false, 21)
% plot_parameters_from_prediction_scheme_1(ts, 264, 50, 28, false)
% plot_preds_without_observed(ts, ...
%     mean(nor_preds_7_days_ahead_from_264_with_100_train_data, 1), ...
%     264, 100, 7, false, 14)
% plot_preds_with_observed(ts, ...
%     mean(nor_test_preds_28_days_ahead_from_264_with_100_train_data, 1), ...
%     264, 100, 28, false, 30)
% plot_parameters_from_prediction_scheme_1(ts, 264, 100, 28, false)


%%%%%%%%%%%%%%%%%%%% Global Forecasts %%%%%%%%%%%%%%%%%%%%

% plot_loss_function(loss_func)

% plot_10_global_CNN_LSTM_Preds(floor(exp(ts_global)), ...
%     glob_test_preds_28_days_ahead_from_196_with_196_train_data, ...
%     196, 196, 28, true)
% plot_preds_sarima_gandalf(ts_global, 196, 196, 7, true)
% plot_acf_of_res(ts_global, 196, 196, false)
% plot_acf_of_res(ts_global, 196, 196, true)
%
% plot_preds_without_observed(ts_global, ...
%     mean(glob_preds_7_days_ahead_from_196_with_196_train_data,1), ...
%     196, 196, 7, true, 14)
% plot_preds_with_observed(ts_global, ...
%     mean(glob_test_preds_28_days_ahead_from_196_with_196_train_data, 1), ...
%     196, 196, 28, true, 30)
% plot_parameters_from_prediction_scheme_1(ts_global, 196, 196, 28, true)
%
% plot_10_global_CNN_LSTM_Preds(floor(exp(ts_global)), ...
%     glob_preds_28_days_ahead_from_196_with_196_train_data, ...
%     196, 196, 28, true)
% plot_preds_without_observed(ts_global, ...
%     mean(glob_preds_28_days_ahead_from_196_with_196_train_data, 1), ...
%     196, 196, 28, true, 30)

```

```

%%%%%%%%%%%%%%%%%%%%% Results for table 2 %%%%%%%%%%%%%%%%%%%%


% Generate all numbers to be inserted in table
% Norway:
% extract_percision_results_without_observed(ts, ...
    nor_preds_7_days_ahead_from_100_with_50_train_data, 100, 50, 7)
% extract_percision_results_without_observed(ts, ...
    nor_preds_7_days_ahead_from_100_with_100_train_data, 100, 100, 7)
%
% extract_percision_results_without_observed(ts, ...
    nor_preds_7_days_ahead_from_264_with_50_train_data, 264, 50, 7)
% extract_percision_results_without_observed(ts, ...
    nor_preds_7_days_ahead_from_264_with_100_train_data, 264, 100, 7)
% extract_percision_results_without_observed(ts, ...
    nor_preds_7_days_ahead_from_264_with_264_train_data, 264, 264, 7)
%
% extract_percision_results_without_observed(ts, ...
    nor_preds_7_days_ahead_from_300_with_50_train_data, 300, 50, 7)
% extract_percision_results_without_observed(ts, ...
    nor_preds_7_days_ahead_from_300_with_100_train_data, 300, 100, 7)
% extract_percision_results_without_observed(ts, ...
    nor_preds_7_days_ahead_from_300_with_300_train_data, 300, 300, 7)
%
% extract_percision_results_without_observed(ts, ...
    nor_preds_7_days_ahead_from_500_with_50_train_data, 500, 50, 7)
% extract_percision_results_without_observed(ts, ...
    nor_preds_7_days_ahead_from_500_with_100_train_data, 500, 100, 7)
% extract_percision_results_without_observed(ts, ...
    nor_preds_7_days_ahead_from_500_with_500_train_data, 500, 500, 7)
%
% extract_percision_results_without_observed(ts, ...
    nor_preds_7_days_ahead_from_696_with_50_train_data, 696, 50, 7)
% extract_percision_results_without_observed(ts, ...
    nor_preds_7_days_ahead_from_696_with_100_train_data, 696, 100, 7)
% extract_percision_results_without_observed(ts, ...
    nor_preds_7_days_ahead_from_696_with_696_train_data, 696, 696, 7)
%
%
%
% extract_percision_results_with_observed(ts, ...
    nor_test_preds_28_days_ahead_from_100_with_50_train_data, 100, 50, 28)
% extract_percision_results_with_observed(ts, ...
    nor_test_preds_28_days_ahead_from_100_with_100_train_data, 100, 100, 28)
%
% extract_percision_results_with_observed(ts, ...
    nor_test_preds_28_days_ahead_from_264_with_50_train_data, 264, 50, 28)
% extract_percision_results_with_observed(ts, ...
    nor_test_preds_28_days_ahead_from_264_with_100_train_data, 264, 100, 28)
% extract_percision_results_with_observed(ts, ...
    nor_test_preds_28_days_ahead_from_264_with_264_train_data, 264, 264, 28)
%
% extract_percision_results_with_observed(ts, ...
    nor_test_preds_28_days_ahead_from_300_with_50_train_data, 300, 50, 28)
% extract_percision_results_with_observed(ts, ...
    nor_test_preds_28_days_ahead_from_300_with_100_train_data, 300, 100, 28)
% extract_percision_results_with_observed(ts, ...
    nor_test_preds_28_days_ahead_from_300_with_300_train_data, 300, 300, 28)
%
% extract_percision_results_with_observed(ts, ...
    nor_test_preds_28_days_ahead_from_500_with_50_train_data, 500, 50, 28)
% extract_percision_results_with_observed(ts, ...
    nor_test_preds_28_days_ahead_from_500_with_100_train_data, 500, 100, 28)
% extract_percision_results_with_observed(ts, ...
    nor_test_preds_28_days_ahead_from_500_with_500_train_data, 500, 500, 28)
%
% extract_percision_results_with_observed(ts, ...
    nor_test_preds_28_days_ahead_from_696_with_50_train_data, 696, 50, 28)
% extract_percision_results_with_observed(ts, ...
    nor_test_preds_28_days_ahead_from_696_with_100_train_data, 696, 100, 28)
% extract_percision_results_with_observed(ts, ...
    nor_test_preds_28_days_ahead_from_696_with_696_train_data, 696, 696, 28)
```

```
%
% Global:
% extract_percision_results_without_observed(ts_global, ...
    glob_preds_7_days_ahead_from_100_with_50_train_data, 100, 50, 7)
% extract_percision_results_without_observed(ts_global, ...
    glob_preds_7_days_ahead_from_100_with_100_train_data, 100, 100, 7)
%
% extract_percision_results_without_observed(ts_global, ...
    glob_preds_7_days_ahead_from_196_with_50_train_data, 196, 50, 7)
% extract_percision_results_without_observed(ts_global, ...
    glob_preds_7_days_ahead_from_196_with_100_train_data, 196, 100, 7)
% extract_percision_results_without_observed(ts_global, ...
    glob_preds_7_days_ahead_from_196_with_196_train_data, 196, 196, 7)
%
% extract_percision_results_without_observed(ts_global, ...
    glob_preds_7_days_ahead_from_300_with_50_train_data, 300, 50, 7)
% extract_percision_results_without_observed(ts_global, ...
    glob_preds_7_days_ahead_from_300_with_100_train_data, 300, 100, 7)
% extract_percision_results_without_observed(ts_global, ...
    glob_preds_7_days_ahead_from_300_with_300_train_data, 300, 300, 7)
%
% extract_percision_results_without_observed(ts_global, ...
    glob_preds_7_days_ahead_from_500_with_50_train_data, 500, 50, 7)
% extract_percision_results_without_observed(ts_global, ...
    glob_preds_7_days_ahead_from_500_with_100_train_data, 500, 100, 7)
% extract_percision_results_without_observed(ts_global, ...
    glob_preds_7_days_ahead_from_500_with_500_train_data, 500, 500, 7)
%
% extract_percision_results_without_observed(ts_global, ...
    glob_preds_7_days_ahead_from_743_with_50_train_data, 743, 50, 7)
% extract_percision_results_without_observed(ts_global, ...
    glob_preds_7_days_ahead_from_743_with_100_train_data, 743, 100, 7)
% extract_percision_results_without_observed(ts_global, ...
    glob_preds_7_days_ahead_from_743_with_743_train_data, 743, 743, 7)
%
%
%
% extract_percision_results_with_observed(ts_global, ...
    glob_test_preds_28_days_ahead_from_100_with_50_train_data, 100, 50, 28)
% extract_percision_results_with_observed(ts_global, ...
    glob_test_preds_28_days_ahead_from_100_with_100_train_data, 100, 100, 28)
%
% extract_percision_results_with_observed(ts_global, ...
    glob_test_preds_28_days_ahead_from_196_with_50_train_data, 196, 50, 28)
% extract_percision_results_with_observed(ts_global, ...
    glob_test_preds_28_days_ahead_from_196_with_100_train_data, 196, 100, 28)
% extract_percision_results_with_observed(ts_global, ...
    glob_test_preds_28_days_ahead_from_196_with_196_train_data, 196, 196, 28)
%
% extract_percision_results_with_observed(ts_global, ...
    glob_test_preds_28_days_ahead_from_300_with_50_train_data, 300, 50, 28)
% extract_percision_results_with_observed(ts_global, ...
    glob_test_preds_28_days_ahead_from_300_with_100_train_data, 300, 100, 28)
% extract_percision_results_with_observed(ts_global, ...
    glob_test_preds_28_days_ahead_from_300_with_300_train_data, 300, 300, 28)
%
% extract_percision_results_with_observed(ts_global, ...
    glob_test_preds_28_days_ahead_from_500_with_50_train_data, 500, 50, 28)
% extract_percision_results_with_observed(ts_global, ...
    glob_test_preds_28_days_ahead_from_500_with_100_train_data, 500, 100, 28)
% extract_percision_results_with_observed(ts_global, ...
    glob_test_preds_28_days_ahead_from_500_with_500_train_data, 500, 500, 28)
% %
% extract_percision_results_with_observed(ts_global, ...
    glob_test_preds_28_days_ahead_from_743_with_50_train_data, 743, 50, 28)
% extract_percision_results_with_observed(ts_global, ...
    glob_test_preds_28_days_ahead_from_743_with_100_train_data, 743, 100, 28)
% extract_percision_results_with_observed(ts_global, ...
    glob_test_preds_28_days_ahead_from_743_with_743_train_data, 743, 743, 28)



%%%%%%%%%%%%%%%%%%%%%% Study table with bar plots %%%%%%%%%%%%%%%%%%%%%%
```

```
% Gather measures from Table 2 to plot summarized results:
% RRMSE:
results_RRMSE_28_day_preds = [74 75 73; 60 262 53; 12 10 18; 11 11 25;
    11 11 28; 25 27 29; 25 24 17; 28 28 26; 14 15 30; 15 15 35; 16 16 19;
    11 9 19; 9 9 30; 9 9 46; 7 7 7; 6 6 10; 4 5 5; 4 4 4; 4 4 4; 5 7 12;
    5 5 9; 6 6 11; 4 6 9; 5 5 15; 5 5 39; 9 7 21; 7 7 21; 7 7 85];

results_RRMSE_7_day_preds = [119 119 64; 87 87 65; 17 6 8; 16 15 11;
    12 12 12; 17 17 14; 19 20 7; 16 16 18; 10 14 8; 5 8 12;
    21 21 11; 9 8 23; 9 11 24; 13 4 24;

    30 35 11; 16 16 14; 7 6 8; 4 4 6; 4 4 5; 7 5 11; 7 7 8; 7 7 9; 4 10 13;
    6 7 20; 3 3 18; 15 10 12; 12 14 10; 12 13 12];

input_bar_plot_RRMSE_days_ahead = [mean(results_RRMSE_28_day_preds, 1); ...
    mean(results_RRMSE_7_day_preds, 1)];

results_RRMSE_Norway = [results_RRMSE_7_day_preds(1:14, :); ...
    results_RRMSE_28_day_preds(1:14, :)];
results_RRMSE_Global = [results_RRMSE_7_day_preds(15:28, :); ...
    results_RRMSE_28_day_preds(15:28, :)];

input_bar_plot_RRMSE_data_set = [mean(results_RRMSE_Norway, 1); ...
    mean(results_RRMSE_Global, 1)];

results_RRMSE_full_data = ...
    [results_RRMSE_7_day_preds([1, 3, 6, 9, 12, 15, 17, 20, 23, 26], :);
    results_RRMSE_28_day_preds([1, 3, 6, 9, 12, 15, 17, 20, 23, 26], :)];

results_RRMSE_100_data = ...
    [results_RRMSE_7_day_preds([1, 4, 7, 10, 13, 15, 18, 21, 24, 27], :);
    results_RRMSE_28_day_preds([1, 4, 7, 10, 13, 15, 18, 21, 24, 27], :)];

results_RRMSE_50_data = ...
    [results_RRMSE_7_day_preds([2, 5, 8, 11, 14, 16, 19, 22, 25, 28], :);
    results_RRMSE_28_day_preds([2, 5, 8, 11, 14, 16, 19, 22, 25, 28], :)];

input_bar_plot_RRMSE_sample_size = [mean(results_RRMSE_full_data, 1); ...
    mean(results_RRMSE_100_data, 1); mean(results_RRMSE_50_data, 1)];

input_bar_plot_RRMSE_models = mean([results_RRMSE_full_data; ...
    results_RRMSE_100_data; results_RRMSE_50_data],1);

% MAPE:
results_MAPE_7_day_preds = [100 100 88; 57 57 41; 16 6 8; 12 11 11;
    9 9 11; 14 13 12; 15 16 7; 13 13 14; 7 11 8; 4 5 12; 17 17 9; 7 7 22;
    7 9 18; 11 3 23; 28  31 10; 14 14 13; 6 6 6; 3 3 4; 4 4 3; 6 4 10;
    6 6 7; 6 6 8; 3 9 13; 6 6 20; 2 2 18; 12 9 12; 11 11 10; 11 10 11];

results_MAPE_28_day_preds = [61 62 87; 40 118 41; 9 8 17; 9 8 26; 8 8 28;
    22 20 23; 20 19 23; 23 22 21; 11 12 12; 12 11 20; 13 12 23; 11 9 17;
    9 9 31; 9 8 46;

    6 5 6; 5 5 9; 3 4 3; 3 3 3; 3 3 3; 4 5 11; 4 4 8;
    4 4 9; 4 5 8; 4 4 14; 4 4 39; 7 6 20; 6 6 20; 6 5 83];

input_bar_plot_MAPE_days_ahead = [mean(results_MAPE_28_day_preds, 1); ...
    mean(results_MAPE_7_day_preds, 1)];

results_MAPE_Norway = ...
    [results_MAPE_7_day_preds(1:14, :); results_MAPE_28_day_preds(1:14, :)];
results_MAPE_Global = ...
    [results_MAPE_7_day_preds(15:28, :); results_MAPE_28_day_preds(15:28, :)];

input_bar_plot_MAPE_data_set = [mean(results_MAPE_Norway, 1); ...
    mean(results_MAPE_Global, 1)];

results_MAPE_full_data = ...
    [results_MAPE_7_day_preds([1, 3, 6, 9, 12, 15, 17, 20, 23, 26], :);
    results_MAPE_28_day_preds([1, 3, 6, 9, 12, 15, 17, 20, 23, 26], :)];

results_MAPE_100_data = ...
```

```
    [results_MAPE_7_day_preds([1, 4, 7, 10, 13, 15, 18, 21, 24, 27], :);
    results_MAPE_28_day_preds([1, 4, 7, 10, 13, 15, 18, 21, 24, 27], :)];

results_MAPE_50_data = ...
    [results_MAPE_7_day_preds([2, 5, 8, 11, 14, 16, 19, 22, 25, 28], :);
    results_MAPE_28_day_preds([2, 5, 8, 11, 14, 16, 19, 22, 25, 28], :)];

input_bar_plot_MAPE_sample_size = [mean(results_MAPE_full_data, 1); ...
    mean(results_MAPE_100_data, 1); mean(results_MAPE_50_data, 1)];


input_bar_plot_MAPE_models = mean([results_MAPE_full_data; ...
    results_MAPE_100_data; results_MAPE_50_data],1).';

% % Creating histogram for all RRMSEs:
% RRMSEs_from_table = ...
%     [results_RRMSE_28_day_preds; results_RRMSE_7_day_preds];
% RRMSEs_sarima_table = RRMSEs_from_table(:, 1);
% RRMSEs_gandalf_table = RRMSEs_from_table(:, 2);
% RRMSEs_cnn_table = RRMSEs_from_table(:, 3);
% figure
% sar = histogram(RRMSEs_sarima_table, 40, 'FaceColor', [0.4940 0.1840 0.5560]);
% mean(RRMSEs_sarima_table)
% mean_sar = xline(mean(RRMSEs_sarima_table),'--',...
%     'Color', [1 0 0], 'LineWidth',2);
% legend([sar mean_sar], 'SARIMA model','Mean RRMSE')
% xlabel('RRMSE')
% ylabel('Occurrences')
% set(gcf,'color','w')
% set(gca,'FontSize',24)
% % xticks(min(RRMSEs_sarima_table):5:max(RRMSEs_sarima_table))
% figure
% gand = histogram(RRMSEs_gandalf_table, 40, 'FaceColor', [.2, .9, .5]);
% mean(RRMSEs_gandalf_table)
% mean_gand = xline(mean(RRMSEs_gandalf_table),'--',...
%     'Color', [1 0 0], 'LineWidth',2);
% legend([gand mean_gand], 'Gandalf model','Mean RRMSE')
% xlabel('RRMSE')
% ylabel('Occurrences')
% set(gcf,'color','w')
% set(gca,'FontSize',24)
%
% figure
% cnn = histogram(RRMSEs_cnn_table, 40, 'FaceColor', [0 0.4470 0.7410]);
% mean(RRMSEs_cnn_table)
% mean_cnn = xline(mean(RRMSEs_cnn_table),'--',...
%     'Color', [1 0 0], 'LineWidth',2);
% legend([cnn mean_cnn], 'CNN-LSTM model','Mean RRMSE')
% xlabel('RRMSE')
% ylabel('Occurrences')
% set(gcf,'color','w')
% set(gca,'FontSize',24)


% Creating histogram for all MAPEs:
% MAPEs_from_table =  [results_MAPE_28_day_preds; results_MAPE_7_day_preds];
% MAPEs_sarima_table = MAPEs_from_table(:, 1);
% MAPEs_gandalf_table = MAPEs_from_table(:, 2);
% MAPEs_cnn_table = MAPEs_from_table(:, 3);
% figure
% sar = histogram(MAPEs_sarima_table, 40, 'FaceColor', [0.4940 0.1840 0.5560]);
% mean(MAPEs_sarima_table)
% mean_sar = xline(mean(MAPEs_sarima_table),'--',...
%     'Color', [1 0 0], 'LineWidth',2);
% legend([sar mean_sar], 'SARIMA model','Mean MAPE')
% xlabel('MAPE')
% ylabel('Occurrences')
% set(gcf,'color','w')
% set(gca,'FontSize',24)
% % xticks(min(MAPEs_sarima_table):5:max(MAPEs_sarima_table))
% figure
% gand = histogram(MAPEs_gandalf_table, 40, 'FaceColor', [.2, .9, .5]);
% mean(MAPEs_gandalf_table)
```

```
% mean_gand = xline(mean(MAPEs_gandalf_table),'--',...
%     'Color', [1 0 0], 'LineWidth',2);
% legend([gand mean_gand], 'Gandalf model','Mean MAPE')
% xlabel('MAPE')
% ylabel('Occurrences')
% set(gcf,'color','w')
% set(gca,'FontSize',24)
%
% figure
% cnn = histogram(MAPEs_cnn_table, 40, 'FaceColor', [0 0.4470 0.7410]);
% mean(MAPEs_cnn_table)
% mean_cnn = xline(mean(MAPEs_cnn_table),'--',...
%     'Color', [1 0 0], 'LineWidth',2);
% legend([cnn mean_cnn], 'CNN-LSTM model','Mean MAPE')
% xlabel('MAPE')
% ylabel('Occurrences')
% set(gcf,'color','w')
% set(gca,'FontSize',24)



%
% bin_range = min(min(RRMSEs_from_table)):max(max(RRMSEs_from_table));
% counts_sarima = histc(RRMSEs_sarima, bin_range);
% counts_gandalf = histc(RRMSEs_gandalf, bin_range);
% counts_cnn = histc(RRMSEs_cnn, bin_range);
% counts_sarima_gandalf = counts_sarima  + counts_gandalf;
% counts_total = counts_sarima  + counts_gandalf + counts_cnn;
%
% clr = [0 0.4470 0.7410; .2, .9, .5; 0.4940 0.1840 0.5560];
% colormap(clr);
% figure
% cnn = bar(bin_range, counts_total);
% hold on
% gand = bar(bin_range, counts_sarima_gandalf);
% sar = bar(bin_range, counts_sarima);
% hold off
% xlabel('RRMSE')
% ylabel('Occurrences')
% legend([sar, gand, cnn], 'SARIMA model', 'Gandalf model', 'CNN-LSTM model')
% set(gcf,'color','w')
% set(gca,'FontSize',24)
% set(h, {'DisplayName'}, {'SARIMA', 'Gandalf', 'CNN-LSTM'}')


% % Creating histogram for all MAPEs:
% MAPEs_from_table =  [results_MAPE_28_day_preds; results_MAPE_7_day_preds];
% MAPEs_sarima_table = MAPEs_from_table(:, 1);
% MAPEs_gandalf_table = MAPEs_from_table(:, 2);
% MAPEs_cnn_table = MAPEs_from_table(:, 3);
%
% bin_range = min(min(MAPEs_from_table)):max(max(MAPEs_from_table));
% counts_sarima = histc(MAPEs_sarima, bin_range);
% counts_gandalf = histc(MAPEs_gandalf, bin_range);
% counts_cnn = histc(MAPEs_cnn, bin_range);
% counts_sarima_gandalf = counts_sarima  + counts_gandalf;
% counts_total = counts_sarima  + counts_gandalf + counts_cnn;
%
% clr = [0 0.4470 0.7410; .2, .9, .5; 0.4940 0.1840 0.5560];
% colormap(clr);
% figure
% cnn = bar(bin_range, counts_total);
% hold on
% gand = bar(bin_range, counts_sarima_gandalf);
% sar = bar(bin_range, counts_sarima);
% hold off
% xlabel('MAPE')
% ylabel('Occurrences')
% legend([sar, gand, cnn], 'SARIMA model', 'Gandalf model', 'CNN-LSTM model')
% set(gcf,'color','w')
% set(gca,'FontSize',24)
% set(h, {'DisplayName'}, {'SARIMA', 'Gandalf', 'CNN-LSTM'}')
```

```
% Figure tu create bar plots, switch out text and so on depending on the
% spesific plot
% % figure
% h = bar(input_bar_plot_RRMSE_sample_size); % , 'stacked')
% % ylabel('median MAPE')
% ylabel('median RRMSE')
% % xlabel('Model')
% % xticklabels({'SARIMA', 'Gandalf', 'CNN-LSTM'})
% % xlabel('Data set')
% % xticklabels({'Norway', 'Global'})
% % xlabel('Days-ahead forecast')
% % xticklabels({'28 day', '7 day'})
% xlabel('Sample size')
% xticklabels({'All available', '100', '50'})
% set(gcf,'color','w')
% set(gca,'FontSize',24)
% set(h, {'DisplayName'}, {'SARIMA', 'Gandalf', 'CNN-LSTM'}')
% legend()



%%%%%%%%%%%%%%%%%%%%%%%%% Simulation study %%%%%%%%%%%%%%%%%%%%%%

% simulations = simulate_from_Gandalf(ts_global, 28, 196, 1000).';
% plot_simulations(ts_global, simulations.', 28, 196);
% predictions_on_simulations(ts_global, simulations(1:3, :), ...
%     pred_three_preds_on_sims, 28, 196, true);


RRMSEs_cnn = table2array(readtable(...
    'Predictions\RRMSEs_from_simulation_global_28_days_ahead_with_196_train_data_from196.csv'));
MAPEs_cnn = table2array(readtable(...
    'Predictions\MAPEs_from_simulation_global_28_days_ahead_with_196_train_data_from196.csv'));
pred_three_preds_on_sims = table2array(readtable(...
    'Predictions\pred_three_preds_on_sims.csv'));


%The below functions take approximately 20 hours to run with 1000 sims
% [RRMSEs_sarima, MAPEs_sarima, RRMSEs_gandalf, MAPEs_gandalf, ...
%     ma_sim, sma_sim, arch_sim, garch_sim] = ...
%      RRMSE_MAPE_from_simulation_sarima_and_gandalf...
%      (ts_global, simulations, 196, false);
%
% [log_RRMSEs_sarima, log_MAPEs_sarima, ...
%     log_RRMSEs_gandalf, log_MAPEs_gandalf] = ...
%      RRMSE_MAPE_from_simulation_sarima_and_gandalf...
%      (ts_global, simulations, 196, true);
%
%
% % save('RRMSEs_sarima.mat','RRMSEs_sarima');
% % save('MAPEs_sarima.mat','MAPEs_sarima');
% % save('RRMSEs_gandalf.mat','RRMSEs_gandalf');
% % save('MAPEs_gandalf.mat','MAPEs_gandalf');
% %
% % save('log_RRMSEs_gandalf.mat','log_RRMSEs_gandalf');
% % save('log_MAPEs_gandalf.mat','log_MAPEs_gandalf');
%
%
% load('RRMSEs_sarima.mat','RRMSEs_sarima');
% load('MAPEs_sarima.mat','MAPEs_sarima');
% load('RRMSEs_gandalf.mat','RRMSEs_gandalf');
% load('MAPEs_gandalf.mat','MAPEs_gandalf');
% load('log_RRMSEs_gandalf.mat','log_RRMSEs_gandalf');
% load('log_MAPEs_gandalf.mat','log_MAPEs_gandalf');
%
% Plot the results:
% plot_simulated_RRMSE_MAPE(RRMSEs_cnn, MAPEs_cnn)
% plot_simulated_RRMSE_MAPE(RRMSEs_sarima, MAPEs_sarima)
% plot_simulated_RRMSE_MAPE(RRMSEs_gandalf, MAPEs_gandalf)
```

```
% plot_simulated_RRMSE_MAPE(log_RRMSEs_gandalf, log_MAPEs_gandalf)
% plot_simulated_parameters(ts_global, 196, ma_sim(1:200,:), ...
%     sma_sim(1:200,:), arch_sim(1:200,:), garch_sim(1:200,:))

function extract_percision_results_without_observed...
    (ts, preds_cnn, endpoint, len_train, days_ahead)
    % Does not show plot of predictions, only results. This was used to
    % fill out the summarized table.
    sarima = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    gandalf = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    garchmod = garch('Constant',0.001,'GARCHLags',1, 'ARCHLags',1);
    gandalf.Variance = garchmod;

    train_data = ts(endpoint-len_train+1:endpoint);
    fitted_sarima = estimate(sarima, train_data, 'Display','off');
    fitted_gandalf= estimate(gandalf, train_data, 'Display','off');
    preds_sarima= forecast(fitted_sarima, days_ahead, train_data);
    preds_gandalf = forecast(fitted_gandalf, days_ahead, train_data);

    mean_preds_cnn = mean(preds_cnn, 1);
    %mse_cnn = mse_cnn_lstm(ts, mean_preds_cnn, endpoint, len_train, true);

    % convert to normal scale where needed
    ts = floor(exp(ts));
    test_data = ts(endpoint+1:  endpoint + days_ahead);
    preds_sarima = floor(exp(preds_sarima));
    preds_gandalf = floor(exp(preds_gandalf));

    % calculate RRMSE and MAPE for single predictions
    RRMSE_sarima = sqrt(mean((test_data - preds_sarima).^2))/mean(test_data)*100;
    MAPE_sarima = mean(abs(test_data - preds_sarima)./abs(test_data))*100;
    RRMSE_gandalf = sqrt(mean((test_data - preds_gandalf).^2))/mean(test_data)*100;
    MAPE_gandalf = mean(abs(test_data - preds_gandalf)./abs(test_data))*100;
    RRMSE_cnn = sqrt(mean((test_data - mean_preds_cnn.').^2))/mean(test_data)*100;
    MAPE_cnn = mean(abs(test_data - mean_preds_cnn.')./abs(test_data))*100;

    % calculate RRMAE and MAPE for the 10 CNN-LSTM models
    RRMSE = zeros(10, 1);
    MAPE = zeros(10, 1);
    for i = 1:10
        RRMSE(i) = sqrt(mean((test_data.' - preds_cnn(i, ...
            :)).^2))/mean(test_data)*100;
        MAPE(i) = mean(abs(test_data.' - preds_cnn(i, :))./abs(test_data.'))*100;
    end

    mean_RRMSE_cnn = round(mean(RRMSE), 2);
    mean_MAPE_cnn = round(mean(MAPE), 2);

    disp("...........................................................................")
    disp(["Displaying the results from the ", days_ahead,"-day predictions ...
        starting at ", endpoint, ...
        " using ", len_train, " data without test data:"])
    disp(["RRMSE SARIMA: ", round(RRMSE_sarima, 2), ", MAPE SARIMA: ", ...
        round(MAPE_sarima, 2)])
    disp(["RRMSE Gandalf: ", round(RRMSE_gandalf, 2), ", MAPE Gandalf: ", ...
        round(MAPE_gandalf, 2)])
    disp(["RRMSE mean CNN-LSTM: ", round(RRMSE_cnn, 2), ", MAPE mean CNN-LSTM: ", ...
        round(MAPE_cnn, 2)])
    disp(["Mean RRMSE of ten CNN-LSTM models: ", round(mean_RRMSE_cnn, 2), ", ...
        Mean MAPE of ten CNN-LSTM models: ", round(mean_MAPE_cnn, 2)])


end

function extract_percision_results_with_observed...
    (ts, preds_cnn, endpoint, len_train, days_ahead)
    % Does not show plot of predictions, only results. This was used to
    % fill out the summarizinf table.
    sarima = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    gandalf = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    garchmod = garch('Constant',0.001,'GARCHLags',1, 'ARCHLags',1);
    gandalf.Variance = garchmod;
```

```
    preds_sarima = [];
    preds_gandalf = [];

    train_data = ts(endpoint-len_train+1:endpoint);
    for i = 1:days_ahead
        fitted_sarima = estimate(sarima, train_data, 'Display','off');
        fitted_gandalf= estimate(gandalf, train_data, 'Display','off');
        preds_sarima(end+1) = forecast(fitted_sarima, 1, train_data);
        preds_gandalf(end+1) = forecast(fitted_gandalf, 1, train_data);
        train_data = [train_data; ts(endpoint+i)]; % add next observation to ...
            training data
    end

    mean_preds_cnn = mean(preds_cnn, 1);
    %mse_cnn = mse_cnn_lstm(ts, mean_preds_cnn, endpoint, len_train, true);

    % convert to normal scale where needed
    ts = floor(exp(ts));
    test_data = ts(endpoint+1:  endpoint + days_ahead);
    preds_sarima = floor(exp(preds_sarima));
    preds_gandalf = floor(exp(preds_gandalf));

    % calculate RRMSE and MAPE for single predictions
    RRMSE_sarima = sqrt(mean((test_data - preds_sarima.').^2))/mean(test_data)*100;
    MAPE_sarima = mean(abs(test_data - preds_sarima.')./abs(test_data))*100;
    RRMSE_gandalf = sqrt(mean((test_data - preds_gandalf.').^2))/mean(test_data)*100;
    MAPE_gandalf = mean(abs(test_data - preds_gandalf.')./abs(test_data))*100;
    RRMSE_cnn = sqrt(mean((test_data - mean_preds_cnn.').^2))/mean(test_data)*100;
    MAPE_cnn = mean(abs(test_data - mean_preds_cnn.')./abs(test_data))*100;

    % calculate RRMAE and MAPE for the 10 CNN-LSTM models
    RRMSE = zeros(10, 1);
    MAPE = zeros(10, 1);
    for i = 1:10
        RRMSE(i) = sqrt(mean((test_data.' - preds_cnn(i, ...
            :)).^2))/mean(test_data)*100;
        MAPE(i) = mean(abs(test_data.' - preds_cnn(i, :))./abs(test_data.'))*100;
    end

    mean_RRMSE_cnn = round(mean(RRMSE), 2);
    mean_MAPE_cnn = round(mean(MAPE), 2);
    disp("...............................................................................")
    disp(["Displaying the results from the ", days_ahead,"-day predictions ...
        starting at ", ...
        endpoint, " using ", len_train, " data with test data:"])
    disp(["RRMSE SARIMA: ", round(RRMSE_sarima, 2), ", MAPE SARIMA: ", ...
        round(MAPE_sarima, 2)])
    disp(["RRMSE Gandalf: ", round(RRMSE_gandalf, 2), ", MAPE Gandalf: ", ...
        round(MAPE_gandalf, 2)])
    disp(["RRMSE mean CNN-LSTM: ", round(RRMSE_cnn, 2), ", MAPE mean CNN-LSTM: ", ...
        round(MAPE_cnn, 2)])
    disp(["Mean RRMSE of ten CNN-LSTM models: ", round(mean_RRMSE_cnn, 2), ", ...
        Mean MAPE of ten CNN-LSTM models: ", round(mean_MAPE_cnn, 2)])



end

function predictions_on_simulations...
    (ts, simulations, preds_sim_cnn, days_ahead, endpoint, is_global)
% Displays three forecasts on simulated realizations
sarima = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
gandalf = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
garchmod = garch('Constant',0.001,'GARCHLags',1, 'ARCHLags',1);
gandalf.Variance = garchmod;

% simulation was on normal scale
simulations = log(max(simulations, 0.1));
tic
for i = 1:size(simulations,1) % iterate through each simulation

    preds_sim_cnn(i,:)
    train_data = ts(1:endpoint);
```

```
    % Extract params from simulation model
    preds_sarima = [];
    preds_gandalf = [];
    mse_sarima = [];
    mse_gandalf = [];

    % make predictions, iterating through the current simulation
    for j = 1:size(simulations,2)
        fitted_sarima = estimate(sarima, train_data, 'Display','off');
        fitted_gandalf= estimate(gandalf, train_data, 'Display','off');
        [preds_sarima(end+1), mse_sarima(end+1)] = forecast(fitted_sarima, 1, ...
            train_data);
        [preds_gandalf(end+1), mse_gandalf(end+1)] = forecast(fitted_gandalf, 1, ...
            train_data);
        train_data = [train_data; simulations(i,j)]; % add next simulation to ...
            training data
    end
    % revert predictions and simulations back to normal scale
    upper_sarima = preds_sarima + 1.96*sqrt(mse_sarima);
    lower_sarima = preds_sarima - 1.96*sqrt(mse_sarima);
    upper_gandalf = preds_gandalf + 1.96*sqrt(mse_gandalf);
    lower_gandalf =  preds_gandalf - 1.96*sqrt(mse_gandalf);

    % convert to normal scale where needed
    simulations(i,:) = floor(exp(simulations(i,:)));
    preds_sarima = floor(exp(preds_sarima));preds_gandalf = ...
        floor(exp(preds_gandalf));
    upper_sarima = floor(exp(upper_sarima));lower_sarima = floor(exp(lower_sarima));
    upper_gandalf = floor(exp(upper_gandalf));lower_gandalf = ...
        floor(exp(lower_gandalf));

    plot_length = 21;

    if is_global
        dates_train = index_to_date_global(endpoint-plot_length+1:endpoint);
        dates_test = index_to_date_global(endpoint:endpoint+days_ahead-1);
    else
        dates_train = index_to_date_norway(endpoint-plot_length+1:endpoint);
        dates_test = index_to_date_norway(endpoint:endpoint+days_ahead-1);
    end

    RRMSE_sarima = sqrt(mean((simulations(i,:) - ...
        preds_sarima).^2))/mean(simulations(i,:))*100;
    MAPE_sarima = mean(abs(simulations(i,:) - ...
        preds_sarima)./abs(simulations(i,:)))*100;
    RRMSE_gandalf = sqrt(mean((simulations(i,:) - ...
        preds_gandalf).^2))/mean(simulations(i,:))*100;
    MAPE_gandalf = mean(abs(simulations(i,:) - ...
        preds_gandalf)./abs(simulations(i,:)))*100;
    RRMSE_cnn = sqrt(mean((simulations(i,:) - ...
        preds_sim_cnn(i,:)).^2))/mean(simulations(i,:))*100;
    MAPE_cnn = mean(abs(simulations(i,:) - ...
        preds_sim_cnn(i,:))./abs(simulations(i,:)))*100;


    figure
    hold on
    data = plot(dates_train, ...
        floor(exp(ts(endpoint-plot_length+1:endpoint))),'Color',[0.25, 0.25, 0.25]);
    obs = plot(dates_test,simulations(i,:), 'Color', '#A2142F','LineWidth',3);
    predictions_sarima = plot(dates_test,preds_sarima, 'Color', [0.4940 0.1840 ...
        0.5560], 'LineWidth',2);
    u_sarima = plot(dates_test,upper_sarima, '--','Color', [0.4940 0.1840 ...
        0.5560], 'LineWidth',1);
    l_sarima = plot(dates_test,lower_sarima, '--','Color', [0.4940 0.1840 ...
        0.5560], 'LineWidth',1);
    predictions_gandalf = plot(dates_test,preds_gandalf, 'Color', [.2, .9, .5], ...
        'LineWidth',2);
    u_gandalf = plot(dates_test,upper_gandalf, '--', 'Color', [.2, .9, .5], ...
        'LineWidth',1);
    l_gandalf = plot(dates_test,lower_gandalf, '--', 'Color', [.2, .9, .5], ...
        'LineWidth',1);
```

```
    cnn = plot(dates_test,preds_sim_cnn(i,:), 'Color', [0 0.4470 0.7410], ...
        'LineWidth',2);
    % gtext(['RRMSE sarima: ', num2str(RRMSE_sarima), newline,'RRMSE gandalf: ', ...
        num2str(RRMSE_gandalf)],'FontSize', 24)

    gtext([ ...
        '\color[rgb]{' sprintf('%f,%f,%f', [0.4940 0.1840 0.5560] ) '} SARIMA ...
            model: RRMSE = ', num2str(round(RRMSE_sarima, 2)), '%', ...
        ', MAPE = ', num2str(round(MAPE_sarima, 2)), '%',newline, ...
        '\color[rgb]{' sprintf('%f,%f,%f', [.2, .9, .5]) '} Gandalf model: RRMSE ...
            = ', num2str(round(RRMSE_gandalf, 2)), '%', ...
        ', MAPE = ', num2str(round(MAPE_gandalf, 2)), '%', newline, ...
        '\color[rgb]{' sprintf('%f,%f,%f', [0 0.4470 0.7410]) '} CNN-LSTM model: ...
            RRMSE = ', num2str(round(RRMSE_cnn, 2)), '%', ...
        ', MAPE = ', num2str(round(MAPE_cnn, 2)), '%',
        ], 'Interpreter', 'tex','FontSize', 30);

    legend([data, obs, predictions_sarima, u_sarima, predictions_gandalf, ...
        u_gandalf cnn],...
        'Training data', 'Simulated realization', 'Forecast SARIMA','95% interval ...
            SARIMA', ...
        'Forecast Gandalf','95% interval Gandalf', 'Forecast CNN-LSTM', ...
            'Location', 'NorthWest','FontSize', 30)
    ylabel('New cases')
    xlabel('Date')
    set(gcf,'Color','w')
    set(gca,'FontSize',24)
    hold off
    ax = gca;
    if is_global
        ax.YAxis.Exponent = 3;
    else
        ax.YAxis.Exponent = 0;
    end
    hold off
end
toc
end

function plot_simulated_parameters...
    (ts, endpoint, ma_sim, sma_sim, arch_sim, garch_sim)
    %ma_sim = 28x1000 array
    gandalf = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    garchmod = garch('Constant',0.001,'GARCHLags',1, 'ARCHLags',1);
    gandalf.Variance = garchmod;

    [¬, variances]= estimate(gandalf, ts(1:196), 'Display','off');
    SDs = diag(variances).^0.5;
    SDs = SDs([2 3 6 5]).';


    % first element contain parametes fitted on training data only:
    original_ma = repelem(ma_sim(1,1),34);
    upper_ma = original_ma +2*SDs(1);
    lower_ma = original_ma -2*SDs(1);
    original_sma = repelem(sma_sim(1,1),34);
    upper_sma = original_sma +2*SDs(2);
    lower_sma = original_sma -2*SDs(2);
    original_arch = repelem(arch_sim(1,1),34);
    upper_arch = original_arch +2*SDs(3);
    lower_arch = original_arch -2*SDs(3);
    original_garch = repelem(garch_sim(1,1),34);
    upper_garch = original_garch +2*SDs(4);
    lower_garch = original_garch -2*SDs(4);

    % Plotting
    dates_before = index_to_date_global(endpoint-3:endpoint-1);
    dates_sim = index_to_date_global(endpoint:endpoint+27);
    dates_after = index_to_date_global(endpoint+28:endpoint + 30);

    figure
    subplot(2,2,1)
    hold on
```

```
    orig_ma = plot([dates_before dates_sim dates_after], original_ma, '--', ...
        'Color', 'red', 'LineWidth', 2);
    up_ma = plot([dates_before dates_sim dates_after], upper_ma, '--', 'Color', ...
        'black', 'LineWidth', 1);
    low_ma = plot([dates_before dates_sim dates_after], lower_ma, '--', 'Color', ...
        'black', 'LineWidth', 1);
    ma = plot(dates_sim, ma_sim, 'LineWidth', 1, 'Color', 'red');
    ylabel('Estimated \theta_{MA}')
    xlabel('Date')
    set(gcf,'Color','w')
    set(gca,'FontSize',24)
    legend([orig_ma up_ma ma(1)], 'Estimated \theta_{MA} on training data', ...
        'Condidence interval on training data',...
        'Estimated \theta_{MA} on simulations', 'Location', 'northwest');
    hold off

    subplot(2,2,2)
    hold on
    orig_sma = plot([dates_before dates_sim dates_after], original_sma, '--', ...
        'Color', [0 1 0.7], 'LineWidth', 2);
    up_sma = plot([dates_before dates_sim dates_after], upper_sma, '--', 'Color', ...
        'black', 'LineWidth', 1);
    low_sma = plot([dates_before dates_sim dates_after], lower_sma, '--', ...
        'Color', 'black', 'LineWidth', 1);
    sma = plot(dates_sim, sma_sim, 'LineWidth', 1, 'Color', [0 1 0.7]);
    ylabel('Estimated \theta_{SMA}')
    xlabel('Date')
    set(gcf,'Color','w')
    set(gca,'FontSize',24)
    legend([orig_sma up_sma sma(1)], 'Estimated \theta_{SMA} on training data', ...
        'Condidence interval on training data',...
        'Estimated \theta_{SMA} on simulations', 'Location', 'northwest');
    hold off

    subplot(2,2,3)
    hold on
    orig_arch = plot([dates_before dates_sim dates_after], original_arch, '--', ...
        'Color', 'blue', 'LineWidth', 2);
    arch = plot(dates_sim, arch_sim, 'LineWidth', 1, 'Color', 'blue');
    up_arch = plot([dates_before dates_sim dates_after], upper_arch, '--', ...
        'Color', 'black', 'LineWidth', 1);
    low_arch = plot([dates_before dates_sim dates_after], lower_arch, '--', ...
        'Color', 'black', 'LineWidth', 1);
    ylabel('Estimated \alpha_1')
    xlabel('Date')
    set(gcf,'Color','w')
    set(gca,'FontSize',24)
    legend([orig_arch up_arch arch(1)], 'Estimated \alpha_1 on training data', ...
        'Condidence interval on training data',...
        'Estimated \alpha_1 on simulations', 'Location', 'northwest');
    hold off

    subplot(2,2,4)
    hold on
    orig_garch = plot([dates_before dates_sim dates_after], original_garch, '--', ...
        'Color', 'green', 'LineWidth', 2);
    garch_ = plot(dates_sim, garch_sim, 'LineWidth', 1, 'Color', 'green');
    up_garch = plot([dates_before dates_sim dates_after], upper_garch, '--', ...
        'Color', 'black', 'LineWidth', 1);
    low_garch = plot([dates_before dates_sim dates_after], lower_garch, '--', ...
        'Color', 'black', 'LineWidth', 1);
    ylabel('Estimated \beta_1')
    xlabel('Date')
    set(gcf,'Color','w')
    set(gca,'FontSize',24)
    legend([orig_garch up_garch garch_(1)], 'Estimated \beta_1 on training data', ...
        'Condidence interval on training data',...
        'Estimated \beta_1 on simulations', 'Location', 'northwest');
    hold off
end

function plot_simulated_RRMSE_MAPE(RRMSEs, MAPEs)

```

```
% Uncomment the corresponding accuracy measures on the original data set
%     mean_RRMSE_orig = 3.86; % RRMSE on original SARIMA
%     mean_MAPE_orig = 2.89; % MAPE on original SARIMA
%     mean_RRMSE_orig = 4.81; % RRMSE on original Gandalf
%     mean_MAPE_orig = 3.56; % MAPE on original Gandalf
    mean_RRMSE_orig = 0.3574; % RRMSE on original Gandalf log scale
    mean_MAPE_orig = 0.2834; % MAPE on original Gandalf log scale

%     mean_RRMSE_orig = 4.58; % RRMSE on original CNN-LSTM
%     mean_MAPE_orig = 3.21; % MAPE on original CNN-LSTM

    figure
    set(gcf,'color','w')
    subplot(1, 2, 1)
    hold on
    h = histogram(RRMSEs, 100, 'FaceColor', '#00b551', 'EdgeColor','#00b551');
    orig_RRMSE = xline(mean_RRMSE_orig,'--',['Original RRMSE: ', ...
        num2str(mean_RRMSE_orig)], ...
        'Color', [0 0 1], 'LineWidth',2, 'FontSize',16);
    ylabel("Frequency")
    xlabel("RRMSE")
    hold off
    set(gca,'FontSize',24)
    subplot(1, 2, 2)
    hold on
    h = histogram(MAPEs, 100, 'FaceColor', '#ad003d', 'EdgeColor','#ad003d');
    orig_MAPE = xline(mean_MAPE_orig,'--',['Original MAPE: ', ...
        num2str(mean_MAPE_orig)], ...
        'Color', [0 0 1], 'LineWidth',2, 'FontSize',16);
    ylabel("Frequency")
    xlabel("MAPE")
    hold off
    set(gca,'FontSize',24)
end

function [RRMSEs_sarima, MAPEs_sarima, RRMSEs_gandalf, MAPEs_gandalf, ...
    ma_sim, sma_sim, arch_sim, garch_sim] = ...
    RRMSE_MAPE_from_simulation_sarima_and_gandalf(ts, simulations,  endpoint, ...
        log_scale)
% This function predicts one step ahead with simulated data from
% Gandalf model using the SARIMA model and the Gandalf model
sarima = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
gandalf = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
garchmod = garch('Constant',0.001,'GARCHLags',1, 'ARCHLags',1);
gandalf.Variance = garchmod;


RRMSEs_sarima = [];
MAPEs_sarima = [];
RRMSEs_gandalf = [];
MAPEs_gandalf = [];

ma_sim = [];
sma_sim = [];
arch_sim = [];
garch_sim = [];

simulations = log(max(simulations, 0.1));

% RRMSE and MAPE on the real test data, for reference later:
train_data = ts(1:endpoint);
preds_sarima = [];
preds_gandalf = [];

for i = 1:size(simulations,2)
    fitted_sarima = estimate(sarima, train_data, 'Display','off');
    fitted_gandalf= estimate(gandalf, train_data, 'Display','off');
    preds_sarima(end+1) = forecast(fitted_sarima, 1, train_data);
    preds_gandalf(end+1) = forecast(fitted_gandalf, 1, train_data);
    train_data = [train_data; ts(endpoint+i)]; % add next simulation to training data
end
```

```
test_data = ts(endpoint+1:endpoint+size(simulations,2));

% I d : Ta ut parameterverdiene i loopen over

RRMSE_sarima_orig = sqrt(mean((test_data - preds_sarima.').^2))/mean(test_data)*100
MAPE_sarima_orig = mean(abs(test_data - preds_sarima.')./abs(test_data))*100
RRMSE_gandalf_orig = sqrt(mean((test_data - preds_gandalf.').^2))/mean(test_data)*100
MAPE_gandalf_orig = mean(abs(test_data - preds_gandalf.')./abs(test_data))*100


% simulation was on normal scale, need to be on log-scale to fulfill the
% SARIMA models assumptions.

tic
for i = 1:size(simulations,1) % iterate through each simulation
    train_data = ts(1:endpoint);
    preds_sarima = [];
    preds_gandalf = [];

    ma = [];
    sma = [];
    arch = [];
    garch_ = [];
    % make predictions, iterating through the current simulation
    for j = 1:size(simulations,2)
        fitted_sarima = estimate(sarima, train_data, 'Display','off');
        fitted_gandalf= estimate(gandalf, train_data, 'Display','off');

        ma(end+1) = round(cell2mat(fitted_gandalf.MA), 4);
        sma(end+1)  = round(cell2mat(fitted_gandalf.SMA(7)), 4);
        arch(end+1) = round(cell2mat(fitted_gandalf.Variance.ARCH), 4);
        garch_(end+1)  = round(cell2mat(fitted_gandalf.Variance.GARCH), 4);

        preds_sarima(end+1) = forecast(fitted_sarima, 1, train_data);
        preds_gandalf(end+1) = forecast(fitted_gandalf, 1, train_data);
        train_data = [train_data; simulations(i,j)]; % add next simulation to ...
            training data
    end

    if log_scale == false
        % revert predictions and simulations back to normal scale
        preds_sarima = floor(exp(preds_sarima));
        preds_gandalf = floor(exp(preds_gandalf));
        simulations(i,:) = floor(exp(simulations(i,:)));
    end

    % add parameters
    ma_sim = [ma_sim; ma];
    sma_sim = [sma_sim; sma];
    arch_sim = [arch_sim; arch];
    garch_sim = [garch_sim; garch_];

    % calculate RRMSE and MAPE
    RRMSEs_sarima(end+1) = sqrt(mean((simulations(i, :) - ...
        preds_sarima).^2))/mean(simulations(i, :))*100;
    MAPEs_sarima(end+1) = mean(abs(simulations(i, :) - ...
        preds_sarima)./abs(simulations(i, :)))*100;
    RRMSEs_gandalf(end+1) = sqrt(mean((simulations(i, :) - ...
        preds_gandalf).^2))/mean(simulations(i, :))*100;
    MAPEs_gandalf(end+1) = mean(abs(simulations(i, :) - ...
        preds_gandalf)./abs(simulations(i, :)))*100;
    length(MAPEs_gandalf) % To monitor progress
end
toc
end

function plot_simulations(ts, simulations, days_ahead, endpoint)
    gandalf = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    garchmod = garch('Constant',0.001,'GARCHLags',1, 'ARCHLags',1);
    gandalf.Variance = garchmod;

    train_data = ts(1:endpoint);
```

```
%     test_data = floor(exp(ts(endpoint+1:endpoint+28)));
    rng(123) % Set seed for reproducibility
    fitted_model = estimate(gandalf, train_data, 'Display', 'off');
    [prediction, YMSE]= forecast(fitted_model, days_ahead, 'Y0', train_data);
    upper =  prediction + 1.96*sqrt(YMSE);
    lower =  prediction - 1.96*sqrt(YMSE);
    prediction = floor(exp(prediction)); upper = floor(exp(upper)); lower = ...
        floor(exp(lower));

    % Plot
    plot_length = 21;
    dates_train = index_to_date_global(endpoint-plot_length+1:endpoint);
    dates_test = index_to_date_global(endpoint+1:endpoint+days_ahead);
    figure
    hold on
    data = plot(dates_train, ...
        floor(exp(ts(endpoint-plot_length+1:endpoint))),'Color',[0.25, 0.25, 0.25]);
    obs = plot(dates_test,floor(exp(ts(endpoint+1:endpoint+days_ahead))), ...
        'Color', [1, 0, 0],'LineWidth',3);
    pred = plot(dates_test,prediction, 'Color', [.2, .9, .5], 'LineWidth',2);
    up = plot(dates_test,upper, '--', 'Color', [.2, .9, .5], 'LineWidth',1);
    low = plot(dates_test,lower, '--', 'Color', [.2, .9, .5], 'LineWidth',1);
    sims = plot(dates_test,simulations(:,1:3), 'Color', '#A2142F', 'LineWidth',2);
    up_sims = plot(dates_test,prctile(simulations,97.5,2), '--','Color', 'b', ...
        'LineWidth',1);
    low_sims = plot(dates_test,prctile(simulations,2.5,2), '--','Color', 'b', ...
        'LineWidth',1);
    %med_sims = plot(dates_test,median(simulations,2), 'Color', 'b', 'LineWidth',2);
    % Hvis du vil plotte alle med gjennomsiktighet
    %     for i=1:length(sims)
    %     sims(i).Color = [sims(i).Color 0.05];  % alpha=0.1
    %     end
    legend([data obs sims(1) pred up up_sims], 'Training data', 'Test data', ...
        'Simulations', 'Forecast Gandalf', ...
        '95% theoretical interval Gandalf', '95% quantile range simulations', ...
            'Location', 'northwest')
    ylabel('New cases')
    xlabel('Date')
    set(gcf,'color','w')
    ax = gca;
    ax.YAxis.Exponent = 3;
    set(gca,'FontSize',24)
    hold off
end

function simulations = simulate_from_Gandalf...
    (ts, days_ahead, endpoint, amount_simulations)
    gandalf = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    garchmod = garch('Constant',0.001,'GARCHLags',1, 'ARCHLags',1);
    gandalf.Variance = garchmod;

    train_data = ts(1:endpoint);
    rng(123) % Set seed for reproducibility
    fitted_model = estimate(gandalf, train_data, 'Display', 'off');
    [prediction, YMSE]= forecast(fitted_model, days_ahead, 'Y0', train_data);
    upper =  prediction + 1.96*sqrt(YMSE);
    lower =  prediction - 1.96*sqrt(YMSE);
    prediction = floor(exp(prediction));
    upper = floor(exp(upper));
    lower = floor(exp(lower));
    simulations = floor(exp(simulate(fitted_model,...
        days_ahead, 'NumPaths', amount_simulations, 'Y0', train_data)));
    % Save the results. These will be used in Python later
    writematrix(simulations.', ...
        'Matlab\Saved variables\simulations_from_gandalf.csv')
end

function plot_loss_function(loss_func)
    %[minimum, minimum_ind] = min(loss_func); % find minimal loss and its ...
        corresponding epoc
    figure
    hold on
    loss = plot(loss_func, 'LineWidth', 3);
```

```
    yline(loss_func(100), '--', 'Mean loss after 100 epochs', ...
        'Color', 'red', 'LineWidth',2, 'LabelVerticalAlignment', 'top', ...
            'FontSize', 24);
    ylim([-0.001 0.01])
    ylabel('Mean loss')
    xlabel('Epocs')
    set(gcf,'color','w')
    set(gca,'FontSize',24)
    hold off
end

function plot_acf_of_res(ts, endpoint, len_train, garch_noise)
    % plot ACF of residuals and squared residuals

    train_data = ts(endpoint-len_train+1:endpoint);

    mod = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    if garch_noise
        noise = garch('Constant',0.001,'GARCHLags',1, 'ARCHLags',1);
        mod.Variance = noise
    end
    % fit model
    fitted_model = estimate(mod, train_data, 'Display','off');
    residuals = infer(fitted_model, train_data);

    % res = plot(residuals)
    [¬, p_arch] = archtest(residuals, 'Lags', 2);
    %[¬, p_lbq] = lbqtest(residuals);
    figure

    [acf,lags,bounds] = autocorr(residuals);

    stem(lags,acf, 'LineWidth',2); xlabel('Lag'); ylabel('\gamma(k)');

    hold on;
    h = plot(lags,bounds(1)*ones(length(acf),1), '--','LineWidth',2, 'Color',[1 0 ...
        0]);
    h1 = plot(lags,bounds(2)*ones(length(acf),1), '--' ,'LineWidth',2, 'Color',[1 ...
        0 0]);
    title([''])
    %     title(['Sample ACF of residuals of model from ',...
    %         datestr(index_to_date_norway(endpoint-len_train+1)), ' to ' ...
        datestr(index_to_date_norway(endpoint))])
    set(gcf,'color','w')
    set(gca,'FontSize',40)
    figure
    [acf,lags,bounds] = autocorr(residuals.^2);
    set(gca,'FontSize',40)
    stem(lags,acf, 'LineWidth',2); xlabel('Lag'); ylabel('\gamma(k)');
    hold on;
    h = plot(lags,bounds(1)*ones(length(acf),1), '--','LineWidth',2, 'Color',[1 0 ...
        0]);
    h1 = plot(lags,bounds(2)*ones(length(acf),1), '--' ,'LineWidth',2, 'Color',[1 ...
        0 0]);
    gtext(['P-value for Engels ARCH test: ', num2str(p_arch)],'FontSize',40) % , ...
        newline, 'P-value for Ljung-Box Q-test: ', num2str(p_lbq)]
    title([''])
    %     title(['Sample ACF of squared residuals of model from ', ...
    %         datestr(index_to_date_norway(endpoint-len_train+1)), ' to ' ...
        datestr(index_to_date_norway(endpoint))])
    set(gcf,'color','w')
    set(gca,'FontSize',40)
end

function plot_preds_sarima_gandalf...
    (ts, endpoint, len_train, days_ahead, is_global)
    % Directly predicts the next days_ahead days after endpoint
    % These are then plotted and compared to the CNN-LSTM model with the
    % same prediction sceeme

    % ts should be on log-scale

    plot_length = 14;
```

```
sarima = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
gandalf = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
garchmod = garch('Constant',0.001,'GARCHLags',1, 'ARCHLags',1);
gandalf.Variance = garchmod;

preds_sarima = [];
mse_sarima = [];
preds_gandalf = [];
mse_gandalf = [];

train_data = ts(endpoint-len_train+1:endpoint);

[fitted_sarima, ¬, logL_sarima] = estimate(sarima, train_data, 'Display','off');
[fitted_gandalf, ¬, logL_gandalf]= estimate(gandalf, train_data, ...
    'Display','off');
[preds_sarima, mse_sarima] = forecast(fitted_sarima, days_ahead, train_data);
[preds_gandalf, mse_gandalf] = forecast(fitted_gandalf, days_ahead, train_data);

% Display the AICc values for both models
[¬, ¬, AICc_sarima] = aicbic(logL_sarima, 3, len_train);
[¬, ¬, AICc_gandalf] = aicbic(logL_gandalf, 4, len_train);
AICc_sarima = AICc_sarima.aicc
AICc_gandalf = AICc_gandalf.aicc


% get parameter estimates
MA_sarima = cell2mat(fitted_sarima.MA); SMA_sarima = ...
    cell2mat(fitted_sarima.SMA(7));
sigma2 = fitted_sarima.Variance;
round(SMA_sarima, 2)
MA_gandalf = cell2mat(fitted_gandalf.MA); SMA_gandalf = ...
    cell2mat(fitted_gandalf.SMA(7));
GARCH = cell2mat(fitted_gandalf.Variance.GARCH); ARCH = ...
    cell2mat(fitted_gandalf.Variance.ARCH);

% create 95% intervals
upper_sarima = preds_sarima + 1.96*sqrt(mse_sarima);
lower_sarima = preds_sarima - 1.96*sqrt(mse_sarima);
upper_gandalf = preds_gandalf + 1.96*sqrt(mse_gandalf);
lower_gandalf =  preds_gandalf - 1.96*sqrt(mse_gandalf);

% convert to normal scale where needed
ts = floor(exp(ts));
test_data = ts(endpoint+1:  endpoint + days_ahead);
preds_sarima = floor(exp(preds_sarima));preds_gandalf = ...
    floor(exp(preds_gandalf));
upper_sarima = floor(exp(upper_sarima));lower_sarima = floor(exp(lower_sarima));
upper_gandalf = floor(exp(upper_gandalf));lower_gandalf = ...
    floor(exp(lower_gandalf));

% calculate RRMSE and MAPE
RRMSE_sarima = sqrt(mean((test_data - preds_sarima).^2))/mean(test_data)*100;
MAPE_sarima = mean(abs(test_data - preds_sarima)./abs(test_data))*100;
RRMSE_gandalf = sqrt(mean((test_data - preds_gandalf).^2))/mean(test_data)*100;
MAPE_gandalf = mean(abs(test_data - preds_gandalf)./abs(test_data))*100;

% set dates
if is_global
    dates_train = index_to_date_global(endpoint-plot_length+1:endpoint);
    dates_test = index_to_date_global(endpoint+1:endpoint+days_ahead);
else
    dates_train = index_to_date_norway(endpoint-plot_length+1:endpoint);
    dates_test = index_to_date_norway(endpoint+1:endpoint+days_ahead);
end

% plotting
figure
hold on

data = plot(dates_train, ts(endpoint-plot_length+1:endpoint),'Color',[0.25, ...
    0.25, 0.25]);
obs = plot(dates_test,test_data, 'Color', [1, 0, 0],'LineWidth',3);
```

```
    predictions_sarima = plot(dates_test,preds_sarima, 'Color', [0.4940 0.1840 ... 0.5560], 'LineWidth',2);
    u_sarima = plot(dates_test,upper_sarima, '--','Color', [0.4940 0.1840 ... 0.5560], 'LineWidth',1);
    l_sarima = plot(dates_test,lower_sarima, '--','Color', [0.4940 0.1840 ... 0.5560], 'LineWidth',1);
    predictions_gandalf = plot(dates_test,preds_gandalf, 'Color', [.2, .9, .5], ... 'LineWidth',2);
    u_gandalf = plot(dates_test,upper_gandalf, '--', 'Color', [.2, .9, .5], ... 'LineWidth',1);
    l_gandalf = plot(dates_test,lower_gandalf, '--', 'Color', [.2, .9, .5], ... 'LineWidth',1);

    %ylim([0 1700])

    % insert percision results in plot
    set(gca,'FontSize',24)
    gtext([ ...
        '\color[rgb]{' sprintf('%f,%f,%f', [0.4940 0.1840 0.5560] ) '} SARIMA ... model: RRMSE = ', num2str(round(RRMSE_sarima, 2)), '%', ...
        ',  MAPE = ', num2str(round(MAPE_sarima, 2)), '%',newline, ...
        '\theta_{MA} = ', num2str(round(MA_sarima, 2)), ',  \theta_{SMA} = ', ...
        num2str(round(SMA_sarima, 2)), ',  \sigma^{2} = ', num2str(round(sigma2, ... 2)), newline, ...
        '\color[rgb]{' sprintf('%f,%f,%f', [.2, .9, .5]) '} Gandalf model: RRMSE ... = ', num2str(round(RRMSE_gandalf, 2)), '%', ...
        ',  MAPE = ', num2str(round(MAPE_gandalf, 2)), '%', newline, ...
        '\theta_{MA} = ', num2str(round(MA_gandalf, 2)), ',  \theta_{SMA} = ', ...
        num2str(round(SMA_gandalf, 2)), ',  \alpha_{1} = ', num2str(round(ARCH, ... 2)), ...
        ',  \beta_{1} = ', num2str(round(GARCH, 2))], 'Interpreter', ... 'tex','FontSize', 30);

    legend([data, obs, predictions_sarima, u_sarima, predictions_gandalf, ... u_gandalf],...
        'Training data', 'Test data', 'Forecast SARIMA','95% interval SARIMA', ...
        'Forecast Gandalf','95% interval Gandalf',  'NorthWest','FontSize', 30)
    ylabel('New cases','FontSize', 30)
    xlabel('Date','FontSize', 30)

%     if is_global
%         title(['Forecasts of Global data from ', ... datestr(index_to_date_global(endpoint+1)),'FontSize', 20])
%     else
%         title(['Forecasts of Norwegian data from ', ... datestr(index_to_date_norway(endpoint+1))],'FontSize', 20)
%     end
    set(gcf,'color','w')

    ax = gca;


    if is_global
        ax.YAxis.Exponent = 3;
    else
        ax.YAxis.Exponent = 0;
    end
    hold off
end

function plot_parameters_from_prediction_scheme_1...
    (ts, endpoint, len_train, days_ahead, is_global)
    sarima = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    gandalf = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    garchmod = garch('Constant',0.001,'GARCHLags',1, 'ARCHLags',1);
    gandalf.Variance = garchmod;

    preds_sarima = [];
    preds_gandalf = [];

    train_data = ts(endpoint-len_train+1:endpoint);

    % Gather initial SARIMA parameters
```

```
    fitted_sarima = estimate(sarima, train_data, 'Display','off');
    inintial_MA_sarima = round(cell2mat(fitted_sarima.MA), 4);
    inintial_SMA_sarima = round(cell2mat(fitted_sarima.SMA(7)), 4);
    inintial_SIGMA_sarima = round(fitted_sarima.Variance, 4);

    % Gather initial gandalf parameters
    fitted_gandalf = estimate(gandalf, train_data, 'Display','off');
    inintial_MA_gandalf = round(cell2mat(fitted_gandalf.MA), 4);
    inintial_SMA_gandalf = round(cell2mat(fitted_gandalf.SMA(7)), 4);
    inintial_ARCH_gandalf = round(cell2mat(fitted_gandalf.Variance.ARCH), 4);
    inintial_GARCH_gandalf = round(cell2mat(fitted_gandalf.Variance.GARCH), 4);

    preds_sarima = [];
    preds_gandalf = [];

    MA_sarima = [];
    SMA_sarima = [];
    SIGMA_sarima = [];
    MA_upper_sarima = [];
    SMA_upper_sarima = [];
    SIGMA_upper_sarima = [];
    MA_lower_sarima = [];
    SMA_lower_sarima = [];
    SIGMA_lower_sarima = [];
    MA_lower_gandalf = [];
    SMA_lower_gandalf = [];
    ARCH_lower_gandalf = [];
    GARCH_lower_gandalf = [];
    MA_gandalf = [];
    SMA_gandalf = [];
    ARCH_gandalf = [];
    GARCH_gandalf = [];
    MA_upper_gandalf = [];
    SMA_upper_gandalf = [];
    ARCH_upper_gandalf = [];
    GARCH_upper_gandalf = [];
    MA_lower_gandalf = [];
    SMA_lower_gandalf = [];
    ARCH_lower_gandalf = [];
    GARCH_lower_gandalf = [];

    for i = 1:days_ahead
        [fitted_sarima, varmat_sarima] = estimate(sarima, train_data, ...
            'Display','off');
        [fitted_gandalf, varmat_gandalf]= estimate(gandalf, train_data, ...
            'Display','off');

        SDs = diag(varmat_sarima).^0.5;
        SDs = SDs([2 3 4]).';
        SDs = diag(varmat_gandalf).^0.5;
        SDs = SDs([2 3 6 5]).';

        % Collect parameter estimates from SARIMA model
        MA_new_sarima = round(cell2mat(fitted_sarima.MA), 4);
        MA_sarima = [MA_sarima MA_new_sarima];
        MA_upper_sarima = [MA_upper_sarima MA_new_sarima+2*SDs(1)];
        MA_lower_sarima = [MA_lower_sarima  MA_new_sarima-2*SDs(1)];
        SMA_new_sarima = round(cell2mat(fitted_sarima.SMA(7)), 4);
        SMA_sarima = [SMA_sarima SMA_new_sarima];
        SMA_upper_sarima = [SMA_upper_sarima SMA_new_sarima+2*SDs(2)];
        SMA_lower_sarima = [SMA_lower_sarima  SMA_new_sarima-2*SDs(2)];
        SIGMA_new_sarima = round(fitted_sarima.Variance, 4);
        SIGMA_sarima = [SIGMA_sarima SIGMA_new_sarima];
        SIGMA_upper_sarima = [SIGMA_upper_sarima SIGMA_new_sarima+2*SDs(3)];
        SIGMA_lower_sarima = [SIGMA_lower_sarima  SIGMA_new_sarima-2*SDs(4)];

        % Collect parameter estimates from Gandalf model
        MA_new_gandalf = round(cell2mat(fitted_gandalf.MA), 4);
        MA_gandalf = [MA_gandalf MA_new_gandalf];
        MA_upper_gandalf = [MA_upper_gandalf MA_new_gandalf+2*SDs(1)];
        MA_lower_gandalf = [MA_lower_gandalf  MA_new_gandalf-2*SDs(1)];
        SMA_new_gandalf = round(cell2mat(fitted_gandalf.SMA(7)), 4);
        SMA_gandalf = [SMA_gandalf SMA_new_gandalf];
```

```
        SMA_upper_gandalf = [SMA_upper_gandalf SMA_new_gandalf+2*SDs(2)];
        SMA_lower_gandalf = [SMA_lower_gandalf  SMA_new_gandalf-2*SDs(2)];
        ARCH_new_gandalf = round(cell2mat(fitted_gandalf.Variance.ARCH), 4);
        ARCH_gandalf = [ARCH_gandalf ARCH_new_gandalf];
        ARCH_upper_gandalf = [ARCH_upper_gandalf ARCH_new_gandalf+2*SDs(3)];
        ARCH_lower_gandalf = [ARCH_lower_gandalf  ARCH_new_gandalf-2*SDs(4)];
        GARCH_new_gandalf = round(cell2mat(fitted_gandalf.Variance.GARCH), 4);
        GARCH_gandalf = [GARCH_gandalf GARCH_new_gandalf];
        GARCH_upper_gandalf = [GARCH_upper_gandalf GARCH_new_gandalf+2*SDs(4)];
        GARCH_lower_gandalf = [GARCH_lower_gandalf  GARCH_new_gandalf-2*SDs(4)];

        train_data = [train_data; ts(endpoint+i)]; % add next observation to ...
            training data
    end

    if is_global
        dates_test = index_to_date_global(endpoint:endpoint+days_ahead-1);
    else
        dates_test = index_to_date_norway(endpoint:endpoint+days_ahead-1);
    end

    % Plot SARIMA parameters
    figure
    subplot(2,5,[1 2])
    hold on
    ma = plot(dates_test, MA_sarima, 'Color', 'red','LineWidth',2);
    ma_upper = plot(dates_test, MA_upper_sarima,'black:','LineWidth',2);
    ma_lower = plot(dates_test, MA_lower_sarima,'black:','LineWidth',2);
    yline(inintial_MA_sarima, '--', 'Original \theta_{MA} estimate', ...
        'Color', 'red', 'LineWidth',2, 'LabelVerticalAlignment', 'bottom');
    legend([ma, ma_upper, ma_lower], '\theta_{MA}','95% confidence interval')
    ylabel('Estimated \theta_{MA}')
    xlabel('Date')
    set(gcf,'Color','w')
    set(gca,'FontSize',24)
    hold off


    subplot(2,5, [4 5])
    hold on
    sma = plot(dates_test, SMA_sarima, 'Color', [0 1 0.7],'LineWidth',2);
    sma_upper = plot(dates_test, SMA_upper_sarima,'black:','LineWidth',2);
    sma_lower = plot(dates_test, SMA_lower_sarima,'black:','LineWidth',2);
    yline(inintial_SMA_sarima, '--', 'Original \theta_{SMA} estimate', ...
        'Color', [0 1 0.7], 'LineWidth',2, 'LabelVerticalAlignment', 'bottom');
    legend([sma sma_upper], '\theta_{SMA}','95% confidence intervall')
    ylabel('Estimated \theta_{SMA}')
    xlabel('Date')
    set(gcf,'Color','w')
    set(gca,'FontSize',24)
    hold off

    subplot(2,5, [7 8 9])
    hold on
    sigma = plot(dates_test, SIGMA_sarima, 'Color', 'blue','LineWidth',2);
    sigma_upper = plot(dates_test, SIGMA_upper_sarima,'black:','LineWidth',2);
    sigma_lower = plot(dates_test, SIGMA_lower_sarima,'black:','LineWidth',2);
    yline(inintial_SIGMA_sarima, '--', 'Original \sigma^2 estimate', ...
        'Color', 'blue', 'LineWidth',2, 'LabelVerticalAlignment', 'bottom');
    legend([sigma sigma_upper], '\sigma^2','95% confidence intervall')
    ylabel('Estimated \sigma^2')
    xlabel('Date')
    set(gcf,'Color','w')
    set(gca,'FontSize',24)
    hold off

    % Plot Gandalf parameters
    figure
    subplot(2,2,1)
    hold on
    ma = plot(dates_test, MA_gandalf, 'Color', 'red','LineWidth',2);
    ma_upper = plot(dates_test, MA_upper_gandalf,'black:','LineWidth',2);
    ma_lower = plot(dates_test, MA_lower_gandalf,'black:','LineWidth',2);
```

```
    yline(inintial_MA_gandalf, '--', 'Original \theta_{MA} estimate', ...
        'Color', 'red', 'LineWidth',2, 'LabelVerticalAlignment', 'bottom');
    legend([ma, ma_upper, ma_lower], '\theta_{MA}','95% confidence interval')
    ylabel('Estimated \theta_{MA}')
    xlabel('Date')
    set(gcf,'Color','w')
    set(gca,'FontSize',24)
    hold off


    subplot(2,2,2)
    hold on
    sma = plot(dates_test, SMA_gandalf, 'Color', [0 1 0.7],'LineWidth',2);
    sma_upper = plot(dates_test, SMA_upper_gandalf,'black:','LineWidth',2);
    sma_lower = plot(dates_test, SMA_lower_gandalf,'black:','LineWidth',2);
    yline(inintial_SMA_gandalf, '--', 'Original \theta_{SMA} estimate', ...
        'Color', [0 1 0.7], 'LineWidth',2, 'LabelVerticalAlignment', 'bottom');
    legend([sma sma_upper], '\theta_{SMA}','95% confidence interval')
    ylabel('Estimated \theta_{SMA}')
    xlabel('Date')
    set(gcf,'Color','w')
    set(gca,'FontSize',24)
    hold off

    subplot(2,2,3)
    hold on
    arch = plot(dates_test, ARCH_gandalf, 'Color', 'blue','LineWidth',2);
    arch_upper = plot(dates_test, ARCH_upper_gandalf,'black:','LineWidth',2);
    arch_lower = plot(dates_test, ARCH_lower_gandalf,'black:','LineWidth',2);
    yline(inintial_ARCH_gandalf, '--', 'Original \alpha_{1} estimate', ...
        'Color', 'blue', 'LineWidth',2, 'LabelVerticalAlignment', 'bottom');
    legend([arch arch_upper], '\alpha_{1}','95% confidence interval')
    ylabel('Estimated \alpha_{1}')
    xlabel('Date')
    set(gcf,'Color','w')
    set(gca,'FontSize',24)
    hold off

    subplot(2,2,4)
    hold on
    garch_plot = plot(dates_test, GARCH_gandalf, 'Color', 'green','LineWidth',2);
    garch_upper = plot(dates_test, GARCH_upper_gandalf,'black:','LineWidth',2);
    garch_lower = plot(dates_test, GARCH_lower_gandalf,'black:','LineWidth',2);
    yline(inintial_GARCH_gandalf, '--', 'Original \beta_{1} estimate', ...
        'Color', 'green', 'LineWidth',2, 'LabelVerticalAlignment', 'bottom');
    legend([garch_plot garch_upper], '\beta_{1}','95% confidence interval')
    ylabel('Estimated \beta_{1}')
    xlabel('Date')
    set(gcf,'Color','w')
    set(gca,'FontSize',24)
    hold off
end

function plot_all_predictions_with_observed...
    (ts, preds_cnn, endpoint, len_train, days_ahead, is_global)
    % Plots the two previous functions in the same figure
    % ts is on log-scale
    plot_length = 14;
    sarima = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    gandalf = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    garchmod = garch('Constant',0.001,'GARCHLags',1, 'ARCHLags',1);
    gandalf.Variance = garchmod;

    preds_sarima = [];
    mse_sarima = [];
    preds_gandalf = [];
    mse_gandalf = [];

    train_data = ts(endpoint-len_train+1:endpoint);
    for i = 1:days_ahead
        fitted_sarima = estimate(sarima, train_data, 'Display','off');
        fitted_gandalf= estimate(gandalf, train_data, 'Display','off');
        [preds_sarima(end+1), mse_sarima(end+1)] = ...
```

```
            forecast(fitted_sarima, 1, train_data);
        [preds_gandalf(end+1), mse_gandalf(end+1)] = ...
            forecast(fitted_gandalf, 1, train_data);
        % add next observation to training data:
        train_data = [train_data; ts(endpoint+i)];
    end

    mean_preds_cnn = mean(preds_cnn, 1);
    %mse_cnn = mse_cnn_lstm(ts, mean_preds_cnn, endpoint, len_train, true);

    % create 95% intervals
    upper_sarima = preds_sarima + 1.96*sqrt(mse_sarima);
    lower_sarima = preds_sarima - 1.96*sqrt(mse_sarima);
    upper_gandalf = preds_gandalf + 1.96*sqrt(mse_gandalf);
    lower_gandalf =  preds_gandalf - 1.96*sqrt(mse_gandalf);
    %upper_cnn = mean_preds_cnn + 1.96*round(sqrt(mse_cnn));
    %lower_cnn = mean_preds_cnn - 1.96*round(sqrt(mse_cnn));

    % convert to normal scale where needed
    ts = floor(exp(ts));
    test_data = ts(endpoint+1:  endpoint + days_ahead);
    preds_sarima = floor(exp(preds_sarima));preds_gandalf = ...
        floor(exp(preds_gandalf));
    upper_sarima = floor(exp(upper_sarima));lower_sarima = ...
        floor(exp(lower_sarima));
    upper_gandalf = floor(exp(upper_gandalf));lower_gandalf = ...
        floor(exp(lower_gandalf));

    % calculate RRMSE and MAPE for single predictions
    RRMSE_sarima = ...
        sqrt(mean((test_data - preds_sarima).^2))/mean(test_data)*100;
    MAPE_sarima = ...
        mean(abs(test_data - preds_sarima)./abs(test_data))*100;
    RRMSE_gandalf = ...
        sqrt(mean((test_data - preds_gandalf).^2))/mean(test_data)*100;
    MAPE_gandalf = ...
        mean(abs(test_data - preds_gandalf)./abs(test_data))*100;
    RRMSE_cnn = ...
        sqrt(mean((test_data - mean_preds_cnn.').^2))/mean(test_data)*100;
    MAPE_cnn = ...
        mean(abs(test_data - mean_preds_cnn.')./abs(test_data))*100;

    % calculate RRMAE and MAPE for the 10 CNN-LSTM models
    RRMSE = zeros(10, 1);
    MAPE = zeros(10, 1);
    for i = 1:10
        RRMSE(i) = sqrt(mean((test_data.' - preds_cnn(i, :)).^2))/...
            mean(test_data)*100;
        MAPE(i) = mean(abs(test_data.' - preds_cnn(i, :))./...
            abs(test_data.'))*100;
    end

    % set dates
    if is_global
        dates_train = index_to_date_global(endpoint-plot_length+1:endpoint);
        dates_test = index_to_date_global(endpoint+1:endpoint+days_ahead);
    else
        dates_train = index_to_date_norway(endpoint-plot_length+1:endpoint);
        dates_test = index_to_date_norway(endpoint+1:endpoint+days_ahead);
    end
    % generate RGB colors for plot for the 10 CNN-LSTM models
    rng(1234)
    cols = [];
    for i = 1:10
    c = [rand, rand, rand];
    cols = [cols; c];
    end

    % plotting
    figure
    set(gca,'FontSize',24)
    subplot(1,2,1)
    hold on
```

```
    data = plot(dates_train, ts(endpoint-plot_length+1:endpoint),...
        'Color',[0.25, 0.25, 0.25]);
    obs = plot(dates_test, test_data, 'Color', [1 0 0],'LineWidth',3);
    pred1 = plot(dates_test,preds_cnn(1, :), 'Color', cols(1, :), 'LineWidth',1);
    pred2 = plot(dates_test,preds_cnn(2, :), 'Color', cols(2, :), 'LineWidth',1);
    pred3 = plot(dates_test,preds_cnn(3, :), 'Color', cols(3, :), 'LineWidth',1);
    pred4 = plot(dates_test,preds_cnn(4, :), 'Color', cols(4, :), 'LineWidth',1);
    pred5 = plot(dates_test,preds_cnn(5, :), 'Color', cols(5, :), 'LineWidth',1);
    pred6 = plot(dates_test,preds_cnn(6, :), 'Color', cols(6, :), 'LineWidth',1);
    pred7 = plot(dates_test,preds_cnn(7, :), 'Color', cols(7, :), 'LineWidth',1);
    pred8 = plot(dates_test,preds_cnn(8, :), 'Color', cols(8, :), 'LineWidth',1);
    pred9 = plot(dates_test,preds_cnn(9, :), 'Color', cols(9, :), 'LineWidth',1);
    pred10 = plot(dates_test,preds_cnn(10, :), 'Color', cols(10, :), 'LineWidth',1);

    gtext([ ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(1,:) ) '} Model 1: RRMSE = ', ...
        num2str(round(RRMSE(1), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(1), 2)), '%',newline, ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(2,:) ) '} Model 2: RRMSE = ', ...
        num2str(round(RRMSE(2), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(2), 2)), '%',newline, ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(3,:) ) '} Model 3: RRMSE = ', ...
        num2str(round(RRMSE(3), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(3), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(4,:) ) '} Model 4: RRMSE = ', ...
        num2str(round(RRMSE(4), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(4), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(5,:) ) '} Model 5: RRMSE = ', ...
        num2str(round(RRMSE(5), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(5), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(6,:) ) '} Model 6: RRMSE = ', ...
        num2str(round(RRMSE(6), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(6), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(7,:) ) '} Model 7: RRMSE = ', ...
        num2str(round(RRMSE(7), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(7), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(8,:) ) '} Model 8: RRMSE = ', ...
        num2str(round(RRMSE(8), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(8), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(9,:) ) '} Model 9: RRMSE = ', ...
        num2str(round(RRMSE(9), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(9), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(10,:) ) '} Model 10: RRMSE = ', ...
        num2str(round(RRMSE(10), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(10), 2))], 'Interpreter', 'tex','FontSize', 30);

    gtext(['Mean RRMSE = ', num2str(round(mean(RRMSE), 2)), '%',', mean MAPE = ', ...
        num2str(round(mean(MAPE), 2)), '%'],'FontSize', 30)

    legend([data, obs], 'Training data', 'Test data', 'NorthWest','FontSize', 30)
    ylabel('New cases')
    xlabel('Date')
    if is_global
        title(['Forcast with 10 CNN-LSTM models on Global data from ', ...
            datestr(index_to_date_global(endpoint+1)),...
        ' with sample size ', num2str(len_train), ' and nessesary observed ...
            data'],'FontSize', 60)
    else
        title(['Forcast with 10 CNN-LSTM models on Norwegian data from ', ...
            datestr(index_to_date_norway(endpoint+1)),...
        ' with sample size ', num2str(len_train), ' and nessesary observed ...
            data'],'FontSize', 60)
    end
    ax = gca;
    ax.YAxis.Exponent = 3;
    set(gca,'FontSize',20)
    hold off

    subplot(1,2,2)
    hold on
    data = plot(dates_train, ts(endpoint-plot_length+1:endpoint),'Color',[0.25, ...
        0.25, 0.25]);
    obs = plot(dates_test,test_data, 'Color', [1, 0, 0],'LineWidth',3);
```

```
    predictions_sarima = plot(dates_test,preds_sarima, 'Color', [0.4940 0.1840 ...
        0.5560], 'LineWidth',2);
    u_sarima = plot(dates_test,upper_sarima, '--','Color', [0.4940 0.1840 ...
        0.5560], 'LineWidth',1);
    l_sarima = plot(dates_test,lower_sarima, '--','Color', [0.4940 0.1840 ...
        0.5560], 'LineWidth',1);
    predictions_gandalf = plot(dates_test,preds_gandalf, 'Color', [.2, .9, .5], ...
        'LineWidth',2);
    u_gandalf = plot(dates_test,upper_gandalf, '--', 'Color', [.2, .9, .5], ...
        'LineWidth',1);
    l_gandalf = plot(dates_test,lower_gandalf, '--', 'Color', [.2, .9, .5], ...
        'LineWidth',1);
    predictions_cnn = plot(dates_test,mean_preds_cnn, 'Color', [0 0.4470 0.7410], ...
        'LineWidth',2);
    %u_cnn = plot(dates_test,upper_cnn, '--', 'Color', [0 0.4470 0.7410], ...
        'LineWidth',1);
    %l_cnn = plot(dates_test,lower_cnn, '--', 'Color', [0 0.4470 0.7410], ...
        'LineWidth',1);

    ylim([0 800000])

    % insert percision results in plot
    gtext([ ...
        '\color[rgb]{' sprintf('%f,%f,%f', [0.4940 0.1840 0.5560] ) '} SARIMA ...
            model: RRMSE = ', num2str(round(RRMSE_sarima, 2)), '%', ...
        ', MAPE = ', num2str(round(MAPE_sarima, 2)), '%',newline, ...
        '\color[rgb]{' sprintf('%f,%f,%f', [.2, .9, .5]) '} Gandalf model: RRMSE ...
            = ', num2str(round(RRMSE_gandalf, 2)), '%', ...
        ', MAPE = ', num2str(round(MAPE_gandalf, 2)), '%', newline, ...
        '\color[rgb]{' sprintf('%f,%f,%f', [0 0.4470 0.7410]) '} CNN-LSTM model: ...
            RRMSE = ', num2str(round(RRMSE_cnn, 2)), '%', ...
        ', MAPE = ', num2str(round(MAPE_cnn, 2)), '%',
        ], 'Interpreter', 'tex','FontSize', 30);

    legend([data, obs, predictions_sarima, u_sarima, predictions_gandalf, ...
        u_gandalf, predictions_cnn, u_cnn],...
        'Training data', 'Test data', 'Forecast SARIMA','95% interval SARIMA', ...
        'Forecast Gandalf','95% interval Gandalf', 'Forecast CNN-LSTM',  ...
            'NorthWest','FontSize', 30) % '95% interval CNN-LSTM',
    ylabel('New cases','FontSize', 30)
    xlabel('Date','FontSize', 30)
    if is_global
        title(['Forecasts of Global data from ', ...
            datestr(index_to_date_global(endpoint+1)), ...
            ' using only one-step predictions with observed values'],'FontSize', 30)
    else
        title(['Forecasts of Norwegian data from ', ...
            datestr(index_to_date_norway(endpoint+1)), ...
            ' using only one-step predictions with observed values'],'FontSize', 30)
    end
    set(gcf,'color','w')
    set(gca,'FontSize',20)
    ax = gca;
    ax.YAxis.Exponent = 3;
    if is_global
        ax.YAxis.Exponent = 3;
    else
        ax.YAxis.Exponent = 0;
    end
    hold off
end

function plot_all_predictions_without_observed(ts, preds_cnn, endpoint, ...
    len_train, days_ahead, is_global)
    % Plots the two previous functions in the same figure
    % ts is on log-scale
    plot_length = 14;
    sarima = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    gandalf = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    garchmod = garch('Constant',0.001,'GARCHLags',1, 'ARCHLags',1);
    gandalf.Variance = garchmod;

    %[sarima_1, ¬, logL_without] = estimate(mod, ts_train, 'Display','off');
```

```
    preds_sarima = [];
    mse_sarima = [];
    preds_gandalf = [];
    mse_gandalf = [];

    train_data = ts(endpoint-len_train+1:endpoint);

    tic
    fitted_sarima = estimate(sarima, train_data, 'Display','off');
    fitted_gandalf= estimate(gandalf, train_data, 'Display','off');
    [preds_sarima, mse_sarima] = forecast(fitted_sarima, days_ahead, train_data);
    [preds_gandalf, mse_gandalf] = forecast(fitted_gandalf, days_ahead, train_data);
    toc

    mean_preds_cnn = mean(preds_cnn, 1);
    %mse_cnn = mse_cnn_lstm(ts, mean_preds_cnn, endpoint, len_train, false);

    % create 95% intervals
    upper_sarima = preds_sarima + 1.96*sqrt(mse_sarima);
    lower_sarima = preds_sarima - 1.96*sqrt(mse_sarima);
    upper_gandalf = preds_gandalf + 1.96*sqrt(mse_gandalf);
    lower_gandalf =  preds_gandalf - 1.96*sqrt(mse_gandalf);
    %upper_cnn = mean_preds_cnn + 1.96*round(sqrt(mse_cnn));
    %lower_cnn = mean_preds_cnn - 1.96*round(sqrt(mse_cnn));

    % convert to normal scale where needed
    ts = floor(exp(ts));
    test_data = ts(endpoint+1:  endpoint + days_ahead);
    preds_sarima = floor(exp(preds_sarima));preds_gandalf = ...
        floor(exp(preds_gandalf));
    upper_sarima = floor(exp(upper_sarima));lower_sarima = floor(exp(lower_sarima));
    upper_gandalf = floor(exp(upper_gandalf));lower_gandalf = ...
        floor(exp(lower_gandalf));

    % calculate RRMSE and MAPE for single predictions
    RRMSE_sarima = sqrt(mean((test_data - preds_sarima).^2))/mean(test_data)*100;
    MAPE_sarima = mean(abs(test_data - preds_sarima)./abs(test_data))*100;
    RRMSE_gandalf = sqrt(mean((test_data - preds_gandalf).^2))/mean(test_data)*100;
    MAPE_gandalf = mean(abs(test_data - preds_gandalf)./abs(test_data))*100;
    RRMSE_cnn = sqrt(mean((test_data - mean_preds_cnn.').^2))/mean(test_data)*100;
    MAPE_cnn = mean(abs(test_data - mean_preds_cnn.')./abs(test_data))*100;

    % calculate RRMAE and MAPE for the 10 CNN-LSTM models
    RRMSE = zeros(10, 1);
    MAPE = zeros(10, 1);
    for i = 1:10
        RRMSE(i) = sqrt(mean((test_data.' - preds_cnn(i, ...
            :)).^2))/mean(test_data)*100;
        MAPE(i) = mean(abs(test_data.' - preds_cnn(i, :))./abs(test_data.'))*100;
    end

    % set dates
    if is_global
        dates_train = index_to_date_global(endpoint-plot_length+1:endpoint);
        dates_test = index_to_date_global(endpoint+1:endpoint+days_ahead);
    else
        dates_train = index_to_date_norway(endpoint-plot_length+1:endpoint);
        dates_test = index_to_date_norway(endpoint+1:endpoint+days_ahead);
    end


    % generate RGB colors for plot for the 10 CNN-LSTM models
    rng(1234)
    cols = [];
    for i = 1:10
    c = [rand, rand, rand];
    cols = [cols; c];
    end

    % plotting
    figure
    set(gca,'FontSize',24)
```

```
subplot(1,2,1)
hold on
data = plot(dates_train, ts(endpoint-plot_length+1:endpoint),'Color',[0.25, ...
    0.25, 0.25]);
obs = plot(dates_test, test_data, 'Color', [1 0 0],'LineWidth',3);
pred1 = plot(dates_test,preds_cnn(1, :), 'Color', cols(1, :), 'LineWidth',1);
pred2 = plot(dates_test,preds_cnn(2, :), 'Color', cols(2, :), 'LineWidth',1);
pred3 = plot(dates_test,preds_cnn(3, :), 'Color', cols(3, :), 'LineWidth',1);
pred4 = plot(dates_test,preds_cnn(4, :), 'Color', cols(4, :), 'LineWidth',1);
pred5 = plot(dates_test,preds_cnn(5, :), 'Color', cols(5, :), 'LineWidth',1);
pred6 = plot(dates_test,preds_cnn(6, :), 'Color', cols(6, :), 'LineWidth',1);
pred7 = plot(dates_test,preds_cnn(7, :), 'Color', cols(7, :), 'LineWidth',1);
pred8 = plot(dates_test,preds_cnn(8, :), 'Color', cols(8, :), 'LineWidth',1);
pred9 = plot(dates_test,preds_cnn(9, :), 'Color', cols(9, :), 'LineWidth',1);
pred10 = plot(dates_test,preds_cnn(10, :), 'Color', cols(10, :), 'LineWidth',1);

gtext([ ...
'\color[rgb]{' sprintf('%f,%f,%f', cols(1,:) ) '} Model 1: RRMSE = ', ...
    num2str(round(RRMSE(1), 2)), '%', ...
', MAPE = ', num2str(round(MAPE(1), 2)), '%',newline, ...
'\color[rgb]{' sprintf('%f,%f,%f', cols(2,:) ) '} Model 2: RRMSE = ', ...
    num2str(round(RRMSE(2), 2)), '%', ...
', MAPE = ', num2str(round(MAPE(2), 2)), '%',newline, ...
'\color[rgb]{' sprintf('%f,%f,%f', cols(3,:) ) '} Model 3: RRMSE = ', ...
    num2str(round(RRMSE(3), 2)), '%', ...
', MAPE = ', num2str(round(MAPE(3), 2)), '%',newline ...
'\color[rgb]{' sprintf('%f,%f,%f', cols(4,:) ) '} Model 4: RRMSE = ', ...
    num2str(round(RRMSE(4), 2)), '%', ...
', MAPE = ', num2str(round(MAPE(4), 2)), '%',newline ...
'\color[rgb]{' sprintf('%f,%f,%f', cols(5,:) ) '} Model 5: RRMSE = ', ...
    num2str(round(RRMSE(5), 2)), '%', ...
', MAPE = ', num2str(round(MAPE(5), 2)), '%',newline ...
'\color[rgb]{' sprintf('%f,%f,%f', cols(6,:) ) '} Model 6: RRMSE = ', ...
    num2str(round(RRMSE(6), 2)), '%', ...
', MAPE = ', num2str(round(MAPE(6), 2)), '%',newline ...
'\color[rgb]{' sprintf('%f,%f,%f', cols(7,:) ) '} Model 7: RRMSE = ', ...
    num2str(round(RRMSE(7), 2)), '%', ...
', MAPE = ', num2str(round(MAPE(7), 2)), '%',newline ...
'\color[rgb]{' sprintf('%f,%f,%f', cols(8,:) ) '} Model 8: RRMSE = ', ...
    num2str(round(RRMSE(8), 2)), '%', ...
', MAPE = ', num2str(round(MAPE(8), 2)), '%',newline ...
'\color[rgb]{' sprintf('%f,%f,%f', cols(9,:) ) '} Model 9: RRMSE = ', ...
    num2str(round(RRMSE(9), 2)), '%', ...
', MAPE = ', num2str(round(MAPE(9), 2)), '%',newline ...
'\color[rgb]{' sprintf('%f,%f,%f', cols(10,:) ) '} Model 10: RRMSE = ', ...
    num2str(round(RRMSE(10), 2)), '%', ...
', MAPE = ', num2str(round(MAPE(10), 2))], 'Interpreter', 'tex','FontSize', 30);

gtext(['Mean RRMSE = ', num2str(round(mean(RRMSE), 2)), '%',', mean MAPE = ', ...
    num2str(round(mean(MAPE), 2)), '%'],'FontSize', 30)

legend([data, obs], 'Training data', 'Test data', 'NorthWest','FontSize', 30)
ylabel('New cases','FontSize', 30)
xlabel('Date','FontSize', 30)
if is_global
    title(['Forcast with 10 CNN-LSTM models on Global data from ', ...
        datestr(index_to_date_global(endpoint+1)),...
    ' with sample size ', num2str(len_train)],'FontSize', 30)
else
    title(['Forcast with 10 CNN-LSTM models on Norwegian data from ', ...
        datestr(index_to_date_norway(endpoint+1)),...
    ' with sample size ', num2str(len_train)],'FontSize', 30)
end
ax = gca;
ax.YAxis.Exponent = 3;
set(gca,'FontSize',20)
hold off

subplot(1,2,2)
hold on
data = plot(dates_train, ts(endpoint-plot_length+1:endpoint),'Color',[0.25, ...
    0.25, 0.25]);
obs = plot(dates_test,test_data, 'Color', [1, 0, 0],'LineWidth',3);
```

```
    predictions_sarima = plot(dates_test,preds_sarima, 'Color', [0.4940 0.1840 ...
        0.5560], 'LineWidth',2);
    u_sarima = plot(dates_test,upper_sarima, '--','Color', [0.4940 0.1840 ...
        0.5560], 'LineWidth',1);
    l_sarima = plot(dates_test,lower_sarima, '--','Color', [0.4940 0.1840 ...
        0.5560], 'LineWidth',1);
    predictions_gandalf = plot(dates_test,preds_gandalf, 'Color', [.2, .9, .5], ...
        'LineWidth',2);
    u_gandalf = plot(dates_test,upper_gandalf, '--', 'Color', [.2, .9, .5], ...
        'LineWidth',1);
    l_gandalf = plot(dates_test,lower_gandalf, '--', 'Color', [.2, .9, .5], ...
        'LineWidth',1);
    predictions_cnn = plot(dates_test,mean_preds_cnn, 'Color', [0 0.4470 0.7410], ...
        'LineWidth',2);
    %u_cnn = plot(dates_test,upper_cnn, '--', 'Color', [0 0.4470 0.7410], ...
        'LineWidth',1);
    %l_cnn = plot(dates_test,lower_cnn, '--', 'Color', [0 0.4470 0.7410], ...
        'LineWidth',1);

    ylim([0 800000])

    % insert percision results in plot
    gtext([ ...
        '\color[rgb]{' sprintf('%f,%f,%f', [0.4940 0.1840 0.5560] ) '} SARIMA ...
            model: RRMSE = ', num2str(round(RRMSE_sarima, 2)), '%', ...
        ', MAPE = ', num2str(round(MAPE_sarima, 2)), '%',newline, ...
        '\color[rgb]{' sprintf('%f,%f,%f', [.2, .9, .5]) '} Gandalf model: RRMSE ...
            = ', num2str(round(RRMSE_gandalf, 2)), '%', ...
        ', MAPE = ', num2str(round(MAPE_gandalf, 2)), '%', newline, ...
        '\color[rgb]{' sprintf('%f,%f,%f', [0 0.4470 0.7410]) '} CNN-LSTM model: ...
            RRMSE = ', num2str(round(RRMSE_cnn, 2)), '%', ...
        ', MAPE = ', num2str(round(MAPE_cnn, 2)), '%',
        ], 'Interpreter', 'tex','FontSize', 30);

    legend([data, obs, predictions_sarima, u_sarima, predictions_gandalf, ...
        u_gandalf, predictions_cnn],...
        'Training data', 'Test data', 'Forecast SARIMA','95% interval SARIMA', ...
        'Forecast Gandalf','95% interval Gandalf', 'Forecast CNN-LSTM',  ...
            'NorthWest','FontSize', 30)
    ylabel('New cases','FontSize', 30)
    xlabel('Date','FontSize', 30)
    if is_global
        title(['Forecasts of Global data from ', ...
            datestr(index_to_date_global(endpoint+1)),' with sample size ', ...
            num2str(len_train)],'FontSize', 30)
    else
        title(['Forecasts of Norwegian data from ', ...
            datestr(index_to_date_norway(endpoint+1)), ' with sample size ', ...
            num2str(len_train)],'FontSize', 30)
    end
    set(gcf,'color','w')
    ax = gca;
    if is_global
        ax.YAxis.Exponent = 3;
    else
        ax.YAxis.Exponent = 0;
    end
    hold off
end

function plot_10_norway_CNN_LSTM_Preds(ts, preds, endpoint, len_train, ...
    days_ahead, use_observed)
    % Plot the results from all 10 CNN-LSTM models
    % preds = 10 days_ahead forecasts
    % use_observed should be true if observations are used to predict the
    % ensuing day
    plot_length = 14;
    test_data = ts(endpoint+1:  endpoint + days_ahead);
    % calculate RRMSE, MAE and OSRE
    RRMSE = zeros(10, 1);
    MAPE = zeros(10, 1);
    for i = 1:10
        RRMSE(i) = sqrt(mean((test_data.' - preds(i, :)).^2))/mean(test_data)*100;
```

```
        MAPE(i) = mean(abs(test_data.' - preds(i, :))./abs(test_data.'))*100;
    end

    dates_train = index_to_date_norway(endpoint-plot_length+1:endpoint);
    dates_test = index_to_date_norway(endpoint+1:endpoint+days_ahead);


    % generate RGB colors for plot
    rng(1234)
    cols = [];
    for i = 1:10
    c = [rand, rand, rand];
    cols = [cols; c];
    end
    figure
    hold on

    data = plot(dates_train, ts(endpoint-plot_length+1:endpoint),'Color',[0.25, ...
        0.25, 0.25]);
    obs = plot(dates_test, test_data, 'Color', [1 0 0],'LineWidth',3);
    pred1 = plot(dates_test,preds(1, :), 'Color', cols(1, :), 'LineWidth',1);
    pred2 = plot(dates_test,preds(2, :), 'Color', cols(2, :), 'LineWidth',1);
    pred3 = plot(dates_test,preds(3, :), 'Color', cols(3, :), 'LineWidth',1);
    pred4 = plot(dates_test,preds(4, :), 'Color', cols(4, :), 'LineWidth',1);
    pred5 = plot(dates_test,preds(5, :), 'Color', cols(5, :), 'LineWidth',1);
    pred6 = plot(dates_test,preds(6, :), 'Color', cols(6, :), 'LineWidth',1);
    pred7 = plot(dates_test,preds(7, :), 'Color', cols(7, :), 'LineWidth',1);
    pred8 = plot(dates_test,preds(8, :), 'Color', cols(8, :), 'LineWidth',1);
    pred9 = plot(dates_test,preds(9, :), 'Color', cols(9, :), 'LineWidth',1);
    pred10 = plot(dates_test,preds(10, :), 'Color', cols(10, :), 'LineWidth',1);

    set(gca,'FontSize',24)
    gtext([ ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(1,:) ) '} Model 1: RRMSE = ', ...
        num2str(round(RRMSE(1), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(1), 2)), '%',newline, ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(2,:) ) '} Model 2: RRMSE = ', ...
        num2str(round(RRMSE(2), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(2), 2)), '%',newline, ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(3,:) ) '} Model 3: RRMSE = ', ...
        num2str(round(RRMSE(3), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(3), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(4,:) ) '} Model 4: RRMSE = ', ...
        num2str(round(RRMSE(4), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(4), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(5,:) ) '} Model 5: RRMSE = ', ...
        num2str(round(RRMSE(5), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(5), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(6,:) ) '} Model 6: RRMSE = ', ...
        num2str(round(RRMSE(6), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(6), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(7,:) ) '} Model 7: RRMSE = ', ...
        num2str(round(RRMSE(7), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(7), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(8,:) ) '} Model 8: RRMSE = ', ...
        num2str(round(RRMSE(8), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(8), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(9,:) ) '} Model 9: RRMSE = ', ...
        num2str(round(RRMSE(9), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(9), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(10,:) ) '} Model 10: RRMSE = ', ...
        num2str(round(RRMSE(10), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(10), 2))], 'Interpreter', 'tex','FontSize', 30);

    gtext(['Mean RRMSE = ', num2str(round(mean(RRMSE), 2)), '%',', mean MAPE = ', ...
        num2str(round(mean(MAPE), 2)), '%'],'FontSize', 30)

    legend([data, obs], 'Training data', 'Test data', 'NorthWest','FontSize', 30)

    ylabel('New cases','FontSize', 30)
    xlabel('Date','FontSize', 30)
    if use_observed
        title(['Predictions\Forcast with 10 CNN-LSTM models on Norwegian data ...
```

```
            from ', datestr(index_to_date_norway(endpoint+1)),...
        ' using the ', num2str(len_train), ' previous days and ...
            observations'],'FontSize', 30)
    else
        title(['Predictions\Forcast with 10 CNN-LSTM models on Norwegian data ...
            from ', datestr(index_to_date_norway(endpoint+1)),...
        ' using the ', num2str(len_train), ' previous days'],'FontSize', 30)
    end
    set(gcf,'color','w')
    set(gca,'FontSize',20)
    ax = gca;
    ax.YAxis.Exponent = 0;
    hold off
end

function plot_10_global_CNN_LSTM_Preds(ts, preds, endpoint, len_train, ...
    days_ahead, use_observed)
    % Plot the results from all 10 CNN-LSTM models
    % preds = 10 days_ahead forecasts
    % use_observed should be true if observations are used to predict the
    % ensuing day
    plot_length = 30;
    test_data = ts(endpoint+1:  endpoint + days_ahead);
    % calculate RRMSE, MAE and OSRE
    RRMSE = zeros(10, 1);
    MAPE = zeros(10, 1);
    for i = 1:10
        RRMSE(i) = sqrt(mean((test_data.' - preds(i, :)).^2))/mean(test_data)*100;
        MAPE(i) = mean(abs(test_data.' - preds(i, :))./abs(test_data.'))*100;
    end

    dates_train = index_to_date_global(endpoint-plot_length+1:endpoint);
    dates_test = index_to_date_global(endpoint+1:endpoint+days_ahead);


    % generate RGB colors for plot
    rng(1234)
    cols = [];
    for i = 1:10
    c = [rand, rand, rand];
    cols = [cols; c];
    end
    figure
    hold on

    data = plot(dates_train, ts(endpoint-plot_length+1:endpoint),'Color',[0.25, ...
        0.25, 0.25]);
    obs = plot(dates_test, test_data, 'Color', [1 0 0],'LineWidth',3);
    pred1 = plot(dates_test,preds(1, :), 'Color', cols(1, :), 'LineWidth',1);
    pred2 = plot(dates_test,preds(2, :), 'Color', cols(2, :), 'LineWidth',1);
    pred3 = plot(dates_test,preds(3, :), 'Color', cols(3, :), 'LineWidth',1);
    pred4 = plot(dates_test,preds(4, :), 'Color', cols(4, :), 'LineWidth',1);
    pred5 = plot(dates_test,preds(5, :), 'Color', cols(5, :), 'LineWidth',1);
    pred6 = plot(dates_test,preds(6, :), 'Color', cols(6, :), 'LineWidth',1);
    pred7 = plot(dates_test,preds(7, :), 'Color', cols(7, :), 'LineWidth',1);
    pred8 = plot(dates_test,preds(8, :), 'Color', cols(8, :), 'LineWidth',1);
    pred9 = plot(dates_test,preds(9, :), 'Color', cols(9, :), 'LineWidth',1);
    pred10 = plot(dates_test,preds(10, :), 'Color', cols(10, :), 'LineWidth',1);

    set(gca,'FontSize',24)
    gtext([ ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(1,:) ) '} Model 1: RRMSE = ', ...
        num2str(round(RRMSE(1), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(1), 2)), '%',newline, ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(2,:) ) '} Model 2: RRMSE = ', ...
        num2str(round(RRMSE(2), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(2), 2)), '%',newline, ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(3,:) ) '} Model 3: RRMSE = ', ...
        num2str(round(RRMSE(3), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(3), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(4,:) ) '} Model 4: RRMSE = ', ...
        num2str(round(RRMSE(4), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(4), 2)), '%',newline ...
```

```
    '\color[rgb]{' sprintf('%f,%f,%f', cols(5,:) ) '} Model 5: RRMSE = ', ...
        num2str(round(RRMSE(5), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(5), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(6,:) ) '} Model 6: RRMSE = ', ...
        num2str(round(RRMSE(6), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(6), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(7,:) ) '} Model 7: RRMSE = ', ...
        num2str(round(RRMSE(7), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(7), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(8,:) ) '} Model 8: RRMSE = ', ...
        num2str(round(RRMSE(8), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(8), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(9,:) ) '} Model 9: RRMSE = ', ...
        num2str(round(RRMSE(9), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(9), 2)), '%',newline ...
    '\color[rgb]{' sprintf('%f,%f,%f', cols(10,:) ) '} Model 10: RRMSE = ', ...
        num2str(round(RRMSE(10), 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE(10), 2))], 'Interpreter', 'tex','FontSize', 30);

    gtext(['Mean RRMSE = ', num2str(round(mean(RRMSE), 2)), '%',', mean MAPE = ', ...
        num2str(round(mean(MAPE), 2)), '%'],'FontSize', 30)

    legend([data, obs], 'Training data', 'Test data', 'NorthWest','FontSize', 30)

    ylabel('New cases','FontSize', 30)
    xlabel('Date','FontSize', 30)
    if use_observed
        title(['Predictions\Forcast with 10 CNN-LSTM models on Global data from ...
            ', datestr(index_to_date_global(endpoint+1)),...
        ' using the ', num2str(len_train), ' previous days and ...
            observations'],'FontSize', 30)
    else
        title(['Predictions\Forcast with 10 CNN-LSTM models on Global data from ...
            ', datestr(index_to_date_global(endpoint+1)),...
        ' using the ', num2str(len_train), ' previous days'],'FontSize', 30)
    end
    set(gcf,'color','w')
    set(gca,'FontSize',20)
    ax = gca;
    ax.YAxis.Exponent = 3;
    hold off
end

function plot_preds_without_observed(ts, preds_cnn, endpoint, len_train, ...
    days_ahead, is_global, plot_length)
    % Directly predicts the next days_ahead days after endpoint
    % These are then plotted and compared to the CNN-LSTM model with the
    % same prediction sceeme

    % ts should be on log-scale
    % preds_cnn should not use observed values after training to forecast


    sarima = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    gandalf = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    garchmod = garch('Constant',0.001,'GARCHLags',1, 'ARCHLags',1);
    gandalf.Variance = garchmod;

    %[sarima_1, ¬, logL_without] = estimate(mod, ts_train, 'Display','off');

    preds_sarima = [];
    mse_sarima = [];
    preds_gandalf = [];
    mse_gandalf = [];

    train_data = ts(endpoint-len_train+1:endpoint);

    tic
    fitted_sarima = estimate(sarima, train_data, 'Display','off');
    fitted_gandalf= estimate(gandalf, train_data, 'Display','off');
    [preds_sarima, mse_sarima] = forecast(fitted_sarima, days_ahead, train_data);
    [preds_gandalf, mse_gandalf] = forecast(fitted_gandalf, days_ahead, train_data);
    toc
```

```
%mse_cnn = mse_cnn_lstm(ts, preds_cnn, endpoint, len_train, false);

% create 95% intervals
upper_sarima = preds_sarima + 1.96*sqrt(mse_sarima);
lower_sarima = preds_sarima - 1.96*sqrt(mse_sarima);
upper_gandalf = preds_gandalf + 1.96*sqrt(mse_gandalf);
lower_gandalf =  preds_gandalf - 1.96*sqrt(mse_gandalf);
%upper_cnn = preds_cnn + 1.96*round(sqrt(mse_cnn));
%lower_cnn = preds_cnn - 1.96*round(sqrt(mse_cnn));

% convert to normal scale where needed
ts = floor(exp(ts));
test_data = ts(endpoint+1:  endpoint + days_ahead);
preds_sarima = floor(exp(preds_sarima));preds_gandalf = ...
    floor(exp(preds_gandalf));
upper_sarima = floor(exp(upper_sarima));lower_sarima = floor(exp(lower_sarima));
upper_gandalf = floor(exp(upper_gandalf));lower_gandalf = ...
    floor(exp(lower_gandalf));

% calculate RRMSE and MAPE
RRMSE_sarima = sqrt(mean((test_data - preds_sarima).^2))/mean(test_data)*100;
MAPE_sarima = mean(abs(test_data - preds_sarima)./abs(test_data))*100;
RRMSE_gandalf = sqrt(mean((test_data - preds_gandalf).^2))/mean(test_data)*100;
MAPE_gandalf = mean(abs(test_data - preds_gandalf)./abs(test_data))*100;
RRMSE_cnn = sqrt(mean((test_data - preds_cnn.').^2))/mean(test_data)*100;
MAPE_cnn = mean(abs(test_data - preds_cnn.')./abs(test_data))*100;

% set dates
if is_global
    dates_train = index_to_date_global(endpoint-plot_length+1:endpoint);
    dates_test = index_to_date_global(endpoint+1:endpoint+days_ahead);
else
    dates_train = index_to_date_norway(endpoint-plot_length+1:endpoint);
    dates_test = index_to_date_norway(endpoint+1:endpoint+days_ahead);
end

% plotting
figure
set(gca,'FontSize',24)
hold on

data = plot(dates_train, ts(endpoint-plot_length+1:endpoint),'Color',[0.25, ...
    0.25, 0.25]);
obs = plot(dates_test,test_data, 'Color', [1, 0, 0],'LineWidth',3);
predictions_sarima = plot(dates_test,preds_sarima, 'Color', [0.4940 0.1840 ...
    0.5560], 'LineWidth',2);
u_sarima = plot(dates_test,upper_sarima, '--','Color', [0.4940 0.1840 ...
    0.5560], 'LineWidth',1);
l_sarima = plot(dates_test,lower_sarima, '--','Color', [0.4940 0.1840 ...
    0.5560], 'LineWidth',1);
predictions_gandalf = plot(dates_test,preds_gandalf, 'Color', [.2, .9, .5], ...
    'LineWidth',2);
u_gandalf = plot(dates_test,upper_gandalf, '--', 'Color', [.2, .9, .5], ...
    'LineWidth',1);
l_gandalf = plot(dates_test,lower_gandalf, '--', 'Color', [.2, .9, .5], ...
    'LineWidth',1);
predictions_cnn = plot(dates_test,preds_cnn, 'Color', [0 0.4470 0.7410], ...
    'LineWidth',2);
%u_cnn = plot(dates_test,upper_cnn, '--', 'Color', [0 0.4470 0.7410], ...
    'LineWidth',1);
%l_cnn = plot(dates_test,lower_cnn, '--', 'Color', [0 0.4470 0.7410], ...
    'LineWidth',1);

%ylim([0 700000])

% insert percision results in plot
gtext([ ...
    '\color[rgb]{' sprintf('%f,%f,%f', [0.4940 0.1840 0.5560] ) '} SARIMA ...
        model: RRMSE = ', num2str(round(RRMSE_sarima, 2)), '%', ...
    ', MAPE = ', num2str(round(MAPE_sarima, 2)), '%',newline, ...
    '\color[rgb]{' sprintf('%f,%f,%f', [.2, .9, .5]) '} Gandalf model: RRMSE ...
        = ', num2str(round(RRMSE_gandalf, 2)), '%', ...
```

```
        ', MAPE = ', num2str(round(MAPE_gandalf, 2)), '%', newline, ...
        '\color[rgb]{' sprintf('%f,%f,%f', [0 0.4470 0.7410]) '} CNN-LSTM model: ...
            RRMSE = ', num2str(round(RRMSE_cnn, 2)), '%', ...
        ', MAPE = ', num2str(round(MAPE_cnn, 2)), '%',
        ], 'Interpreter', 'tex','FontSize', 30);

    legend([data, obs, predictions_sarima, u_sarima, predictions_gandalf, ...
        u_gandalf, predictions_cnn],...
        'Training data', 'Test data', 'Forecast SARIMA','95% interval SARIMA', ...
        'Forecast Gandalf','95% interval Gandalf', 'Forecast CNN-LSTM',  ...
            'NorthWest','FontSize', 30) % '95% interval CNN-LSTM',
    ylabel('New cases','FontSize', 30)
    xlabel('Date','FontSize', 30)
    if is_global
        title(['Forecasts of Global data from ', ...
            datestr(index_to_date_global(endpoint+1))],'FontSize', 30)
    else
      title(['Forecasts of Norwegian data from ', ...
          datestr(index_to_date_norway(endpoint+1))],'FontSize', 30)
    end
    set(gcf,'color','w')
    set(gca,'FontSize',20)
    ax = gca;

    if is_global
        ax.YAxis.Exponent = 3;
    else
        ax.YAxis.Exponent = 0;

    end
    hold off
end

function plot_preds_with_observed(ts, preds_cnn, endpoint, len_train, days_ahead, ...
    is_global, plot_length)
    % performes a series of one step predictions for the SARIMA model and
    % the Gandalf model, where the next observation is included as train
    % data.
    % These are then plotted and compared to the CNN-LSTM model with the
    % same prediction sceeme

    % ts should be on log-scale
    % preds_cnn shold be a series of one-steps from CNN-LSTM model using
    % observations to predict the nex day.


    sarima = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    gandalf = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    garchmod = garch('Constant',0.001,'GARCHLags',1, 'ARCHLags',1);
    gandalf.Variance = garchmod;

    %[sarima_1, ¬, logL_without] = estimate(mod, ts_train, 'Display','off');

    preds_sarima = [];
    mse_sarima = [];
    preds_gandalf = [];
    mse_gandalf = [];

    train_data = ts(endpoint-len_train+1:endpoint);
    for i = 1:days_ahead
        fitted_sarima = estimate(sarima, train_data, 'Display','off');
        fitted_gandalf= estimate(gandalf, train_data, 'Display','off');
        [preds_sarima(end+1), mse_sarima(end+1)] = forecast(fitted_sarima, 1, ...
            train_data);
        [preds_gandalf(end+1), mse_gandalf(end+1)] = forecast(fitted_gandalf, 1, ...
            train_data);
        train_data = [train_data; ts(endpoint+i)]; % add next observation to ...
            training data
    end
    %mse_cnn = mse_cnn_lstm(ts, preds_cnn, endpoint, len_train, true);

    % create 95% intervals
    upper_sarima = preds_sarima + 1.96*sqrt(mse_sarima);
```

```
    lower_sarima = preds_sarima - 1.96*sqrt(mse_sarima);
    upper_gandalf = preds_gandalf + 1.96*sqrt(mse_gandalf);
    lower_gandalf =  preds_gandalf - 1.96*sqrt(mse_gandalf);
    %upper_cnn = preds_cnn + 1.96*round(sqrt(mse_cnn));
    %lower_cnn = preds_cnn - 1.96*round(sqrt(mse_cnn));

    % convert to normal scale where needed
    ts = floor(exp(ts));
    test_data = ts(endpoint+1:  endpoint + days_ahead);
    preds_sarima = floor(exp(preds_sarima));preds_gandalf = ...
        floor(exp(preds_gandalf));
    upper_sarima = floor(exp(upper_sarima));lower_sarima = floor(exp(lower_sarima));
    upper_gandalf = floor(exp(upper_gandalf));lower_gandalf = ...
        floor(exp(lower_gandalf));

    % calculate RRMSE and MAPE
    RRMSE_sarima = sqrt(mean((test_data - preds_sarima.').^2))/mean(test_data)*100;
    MAPE_sarima = mean(abs(test_data - preds_sarima.')./abs(test_data))*100;
    RRMSE_gandalf = sqrt(mean((test_data - preds_gandalf.').^2))/mean(test_data)*100;
    MAPE_gandalf = mean(abs(test_data - preds_gandalf.')./abs(test_data))*100;
    RRMSE_cnn = sqrt(mean((test_data - preds_cnn.').^2))/mean(test_data)*100;
    MAPE_cnn = mean(abs(test_data - preds_cnn.')./abs(test_data))*100;

    % set dates
    if is_global
        dates_train = index_to_date_global(endpoint-plot_length+1:endpoint);
        dates_test = index_to_date_global(endpoint+1:endpoint+days_ahead);
    else
        dates_train = index_to_date_norway(endpoint-plot_length+1:endpoint);
        dates_test = index_to_date_norway(endpoint+1:endpoint+days_ahead);
    end


    % plotting
    figure
    set(gca,'FontSize',24)
    hold on

    data = plot(dates_train, ts(endpoint-plot_length+1:endpoint),'Color',[0.25, ...
        0.25, 0.25]);
    obs = plot(dates_test,test_data, 'Color', [1, 0, 0],'LineWidth',3);
    predictions_sarima = plot(dates_test,preds_sarima, 'Color', [0.4940 0.1840 ...
        0.5560], 'LineWidth',2);
    u_sarima = plot(dates_test,upper_sarima, '--','Color', [0.4940 0.1840 ...
        0.5560], 'LineWidth',1);
    l_sarima = plot(dates_test,lower_sarima, '--','Color', [0.4940 0.1840 ...
        0.5560], 'LineWidth',1);
    predictions_gandalf = plot(dates_test,preds_gandalf, 'Color', [.2, .9, .5], ...
        'LineWidth',2);
    u_gandalf = plot(dates_test,upper_gandalf, '--', 'Color', [.2, .9, .5], ...
        'LineWidth',1);
    l_gandalf = plot(dates_test,lower_gandalf, '--', 'Color', [.2, .9, .5], ...
        'LineWidth',1);
    predictions_cnn = plot(dates_test,preds_cnn, 'Color', [0 0.4470 0.7410], ...
        'LineWidth',2);
    %u_cnn = plot(dates_test,upper_cnn, '--', 'Color', [0 0.4470 0.7410], ...
        'LineWidth',1);
    %l_cnn = plot(dates_test,lower_cnn, '--', 'Color', [0 0.4470 0.7410], ...
        'LineWidth',1);

    gtext([ ...
        '\color[rgb]{' sprintf('%f,%f,%f', [0.4940 0.1840 0.5560] ) '} SARIMA ...
            model: RRMSE = ', num2str(round(RRMSE_sarima, 2)), '%', ...
        ', MAPE = ', num2str(round(MAPE_sarima, 2)), '%',newline, ...
        '\color[rgb]{' sprintf('%f,%f,%f', [.2, .9, .5]) '} Gandalf model: RRMSE ...
            = ', num2str(round(RRMSE_gandalf, 2)), '%', ...
        ', MAPE = ', num2str(round(MAPE_gandalf, 2)), '%', newline, ...
        '\color[rgb]{' sprintf('%f,%f,%f', [0 0.4470 0.7410]) '} CNN-LSTM model: ...
            RRMSE = ', num2str(round(RRMSE_cnn, 2)), '%', ...
        ', MAPE = ', num2str(round(MAPE_cnn, 2)), '%',
        ], 'Interpreter', 'tex','FontSize', 30);

    legend([data, obs, predictions_sarima, u_sarima, predictions_gandalf, ...
```

```
        u_gandalf, predictions_cnn],...
        'Training data', 'Test data', 'Forecast SARIMA','95% interval SARIMA', ...
        'Forecast Gandalf','95% interval Gandalf', 'Forecast CNN-LSTM',  ...
            'NorthWest','FontSize', 30) % '95% interval CNN-LSTM',
    ylabel('New cases','FontSize', 30)
    xlabel('Date','FontSize', 30)
    if is_global
        title(['Forecasts of Global data from ', ...
            datestr(index_to_date_global(endpoint+1)), ...
            ' using only observed values to predict each ensuing ...
                day'],'FontSize', 30)
    else
        title(['Forecasts of Norwegian data from ', ...
            datestr(index_to_date_norway(endpoint+1)), ...
            ' using only observed values to predict each ensuing ...
                day'],'FontSize', 30)
    end
    set(gcf,'color','w')
    set(gca,'FontSize',20)
    ax = gca;

    if is_global
        ax.YAxis.Exponent = 3;
    else
        ax.YAxis.Exponent = 0;
    end
    hold off
end

function mse = mse_cnn_lstm(ts, preds_cnn, endpoint, len_train, use_observed)
    % simulates one step at a time from gandalf model to see how good the
    % prediction of the cnn-lstm model is if the underlying time series was
    % the gandalg model

    % ts should be on log scale
    days_ahead = length(preds_cnn);
    mod = arima('Constant',0,'D',1,'Seasonality',7,'MALags',1,'SMALags',7);
    garchmod = garch('Constant',0.001,'GARCHLags',1, 'ARCHLags',1);
    mod.Variance = garchmod;
    mse = [];
    if use_observed
        log_preds = log(max(preds_cnn, 0.1)); % transform to log-scale to use ...
            with gandalf model
        train_data = ts(endpoint-len_train+1:endpoint); % have to use predictions ...
            as initial observations for simulation??
        for i = 1:days_ahead
            gandalf_mod = estimate(mod, train_data, 'Display','off');
            sim = floor(exp(simulate(gandalf_mod, 1, 'NumPaths', 1000000, 'Y0', ...
                train_data)));
            mse(end+1) = mean((sim - preds_cnn(i)).^2);
            train_data = [train_data;log_preds(i)]; % add newest prediction to ...
                the reain data
        end
    else % do everything all at once???
        gandalf_mod = estimate(mod, ts(endpoint-len_train+1:endpoint), ...
            'Display','off');
        sim = floor(exp(simulate(gandalf_mod, days_ahead, 'NumPaths', 1000000, ...
            'Y0', ts(endpoint-len_train+1:endpoint))));
        for i = 1:days_ahead
            mse(end+1) = mean((sim(i,:) - preds_cnn(i)).^2);
        end
    end
end

function plot_norway_and_global_time_series_and_transform(ts, ts_global, ...
    include_trans)
    % This function was used to plot both time series and their respective
    % log and difference transofrmations.
    ts = floor(exp(ts));
    ts_global = floor(exp(ts_global));

    dates_norway = index_to_date_norway(1:length(ts));
    dates_global = index_to_date_global(1:length(ts_global));
```

```

    if include_trans
        figure
        subplot(3,2,1)
        plot_norway = plot(dates_norway, ts, 'Col', '#2eff8c');
        title('New Cases in Norway')
        ylabel('New cases')
        xlabel('Date')
        ax = gca;
        ax.YAxis.Exponent = 0;
        set(gca,'FontSize',24)
        subplot(3,2,2)
        plot_global = plot(dates_global, ts_global, 'Col', 'Magenta');
        title('Global New Cases')
        ylabel('New cases')
        xlabel('Date')
        set(gcf,'color','w')
        ax = gca;
        ax.YAxis.Exponent = 3;
        set(gca,'FontSize',24)

        subplot(3,2,3)
        plot_norway = plot(dates_norway, log(max(ts, 0.1)), 'Col', '#2eff8c');
        title('New Cases in Norway')
        ylabel('New cases')
        xlabel('Date')
        ax = gca;
        ax.YAxis.Exponent = 0;
        set(gca,'FontSize',24)

        subplot(3,2,4)
        plot_global = plot(dates_global, log(max(ts_global, 0.1)), 'Col', 'Magenta');
        title('Global New Cases')
        ylabel('New cases')
        xlabel('Date')
        set(gcf,'color','w')
        ax = gca;
        ax.YAxis.Exponent = 0;
        set(gca,'FontSize',24)


        % apply differencing on top of log transform:
        ts_trans = diff(diff(log(max(ts, 0.1)), 1), 7);
        ts_trans_global = diff(diff(log(max(ts_global , 0.1)), 1), 7);
        subplot(3,2,5)
        plot_norway = plot(dates_norway(9:end), ts_trans, 'Col', '#2eff8c'); % ...
            Note that values of the first eight days are removed!
        title('New Cases in Norway')
        ylabel('New cases')
        xlabel('Date')
        ax = gca;
        ax.YAxis.Exponent = 0;
        set(gca,'FontSize',24)
        subplot(3,2,6)
        plot_global = plot(dates_global(9:end), ts_trans_global, 'Col', 'Magenta');
        title('Global New Cases')
        ylabel('New cases')
        xlabel('Date')
        ax = gca;
        ax.YAxis.Exponent = 0;
        set(gcf,'color','w')
        set(gca,'FontSize',24)

    else
        figure
        subplot(2,1,1)
        plot_norway = plot(dates_norway, ts, 'Col', '#2eff8c');
        title('New Cases in Norway')
        ylabel('New cases')
        xlabel('Date')
        ax = gca;
        ax.YAxis.Exponent = 0;
        set(gca,'FontSize',24)
```

```
        subplot(2,1,2)
        plot_global = plot(dates_global, ts_global, 'Col', 'Magenta');
        title('Global New Cases')
        ylabel('New cases')
        xlabel('Date')
        set(gcf,'color','w')
        ax = gca;
        ax.YAxis.Exponent = 3;
        set(gca,'FontSize',24)

    end



end

function date = index_to_date_norway(index)
    % get value in date format from index
    % index one is February 21th 2020 for Norwegian data.
    t0 =  datetime(2020,2,21);
    date = t0 + days(index-1);
end

function date = index_to_date_global(index)
    % get value in date format from index
    % index one is January 4th 2020 for Global data.
    t0 =  datetime(2020,1,4);
    date = t0 + days(index-1);
end
```

Eivind H. Brataas

Norwegian University of
Science and Technology