


# Machine Learning for Orbit Determination

**Student Paper****Author(s):**

Rösch, Christine; [Gou, Junyang](#) 

**Publication date:**

2022-01

**Permanent link:**

<https://doi.org/10.3929/ethz-b-000617317>

**Rights / license:**

[In Copyright - Non-Commercial Use Permitted](#)

Interdisciplinary Project HS 2021  
Machine Learning for Orbit Determination



Institute of Geodesy and Photogrammetry  
Chair of Mathematical and Physical Geodesy  
Chair of Space Geodesy  
ETH Zürich

Christine Rösch  
Junyang Gou

Zürich, January 2022

---

**Supervisors:** Prof. Markus Rothacher  
Prof. Benedikt Soja  
Endrit Shehaj  
Kangkang Chen  
Mostafa Kiani Shahvandi

## Abstract

Precise orbit determination is vital for the increasingly vast number of space objects around the Earth. Moreover, accurate orbit prediction of GNSS satellites is essential for many real-time geodetic applications, including real-time navigation. The typical way to obtain accurate orbit predictions is using physics-based orbit propagators. However, the prediction errors accumulate with time because of insufficient modeling of the changing perturbing forces. Motivated by the rapid expansion of computing power and the considerable data volume of satellite orbits available in recent years, we can apply machine learning (ML) and deep learning (DL) algorithms to assess if they can be used to further reduce orbit errors.

In this study, we focus on the orbit prediction of GNSS constellations. We investigate the potential of using different ML and DL algorithms for improving the accuracy of the ultra-rapid products from IGS. As ground truth we consider the IGS final products, and the differences between the ultra-rapid and final products are computed and serve as targets for the ML/DL methods. In this context, we combine the advantages of physics-based and data-driven ML/DL methods. Since the major errors of GNSS orbits are expected to be caused by the deficiency of solar radiation pressure models, we consider different related parameters as additional features to implicitly model the solar impact, such as the  $C_{0,0}$  terms of global ionosphere maps. In order to accurately model the effect of solar radiation pressure on the radial, along-track and cross-track components of the satellite orbit system, the geometric relation between the Sun, the satellite and the Earth are also considered. Furthermore, the performances of different ML/DL algorithms are compared and discussed. Due to the temporal characteristics of the problem, certain sequential modeling algorithms, such as Long Short-Term Memory and Gated Recurrent Unit, show superiority. Our approach shows promising results with average improvements of over 40% in 3D RMS within the 24-hours prediction interval of the ultra-rapid products.

**Keywords:** Precise orbit determination, GNSS, Machine learning, Deep learning, Time series prediction

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Data</b>	<b>4</b>
2.1	GNSS Data . . . . .	4
2.2	Solar Data . . . . .	5
2.2.1	Solar Flux . . . . .	6
2.2.2	K <sub>p</sub> index . . . . .	6
2.2.3	C <sub>0,0</sub> term of ionosphere . . . . .	6
2.2.4	Solar Position . . . . .	7
<b>3</b>	<b>Methodology</b>	<b>8</b>
3.1	Overview . . . . .	8
3.2	Data Preprocessing . . . . .	9
3.2.1	Generate Orbit Errors . . . . .	9
3.2.2	Feature Standardization . . . . .	10
3.3	Machine Learning Algorithms . . . . .	12
3.3.1	Linear Regression . . . . .	13
3.3.2	Decision Tree . . . . .	14
3.3.3	Random Forest . . . . .	14
3.3.4	Support Vector Machine . . . . .	15
3.3.5	Multi Layer Perceptron . . . . .	17
3.4	Deep Learning Algorithms . . . . .	18
3.4.1	Deep Neural Network . . . . .	18
3.4.2	Convolutional Neural Network . . . . .	19

3.4.3	Recurrent Neural Network . . . . .	20
3.4.3.1	Long Short-Term Memory . . . . .	22
3.4.3.2	Gated Recurrent Unit . . . . .	23
3.4.3.3	Bidirectional structure . . . . .	24
3.4.3.4	Encoder-Decoder structure . . . . .	26
3.4.4	Combination of CNN and RNN . . . . .	27
3.4.5	Optimization strategy . . . . .	28
3.5	Feature Inclusion and Grouping . . . . .	30
3.5.1	Feature Inclusion . . . . .	30
3.5.2	Grouping . . . . .	31
3.6	Evaluation metrics . . . . .	32
<b>4</b>	<b>Results and Discussion</b>	<b>33</b>
4.1	Machine Learning . . . . .	33
4.1.1	Results Overview . . . . .	33
4.1.2	Discussion about Linear Regression . . . . .	36
4.1.3	Discussion about Decision Tree . . . . .	37
4.1.4	Discussion about Random Forest . . . . .	38
4.1.5	Discussion about Support Vector Machine . . . . .	38
4.1.6	Discussion about Multi Layer Perceptron . . . . .	39
4.1.7	Discussion about Feature Inclusion and Grouping . . . . .	40
4.2	Deep Learning . . . . .	41
4.2.1	Results Overview . . . . .	41
4.2.2	Discussion about RNN . . . . .	44
4.2.3	Influences of additional features . . . . .	47
4.2.4	Other GNSS constellations . . . . .	51
<b>5</b>	<b>Conclusion and Outlook</b>	<b>54</b>
5.1	Conclusion . . . . .	54
5.2	Outlook . . . . .	55

<b>A</b>	<b>Supplementary Materials</b>	<b>56</b>
A.1	Absolute improvements of individual samples . . . . .	56
A.1.1	Machine learning . . . . .	56
A.1.2	Deep learning . . . . .	57
<b>B</b>	<b>Author Contributions</b>	<b>59</b>
B.1	Christine Rösch . . . . .	59
B.2	Junyang Gou . . . . .	59

# List of Figures

2.1	Schematic diagram of the IGS Ultra-rapid orbit products provided by GFZ . . .	5
3.1	Graphical representation of the pipeline of the study . . . . .	9
3.2	The mean orbit errors of the GPS constellations with $1\text{-}\sigma$ interval (up) and the zoom-in version of the mean orbit errors (down). . . . .	11
3.3	Selected individual samples of R, T, N errors (up) and corresponding standardized R, T, N errors (down) of the GPS constellation. . . . .	12
3.4	Concept of Linear Regression (Goodfellow et al., 2016) . . . . .	13
3.5	Visualization of Decision Tree model (Das, Nipa, 2020) . . . . .	14
3.6	Visualization of Random Forest model (Das, Nipa, 2020) . . . . .	15
3.7	Visualization of Support Vector Machine model (Sharp, Tom, 2021) . . . . .	16
3.8	Visualization of Multi Layer Perceptron model (Avinash Navlani, 2020) . . . .	17
3.9	The schematic representation of a TCN model (Bai et al., 2018). . . . .	19
3.10	The schematic representation of the used CNN model in this study. . . . .	20
3.11	Comparison between the sequence-to-sequence structure and sequence-to-one structure of RNN. . . . .	21
3.12	The schematic representation of a LSTM neuron (Colah, 2015) . . . . .	23
3.13	The schematic representation of a GRU neuron (Colah, 2015) . . . . .	24
3.14	The schematic representation of a bidirectional RNN structure. . . . .	25
3.15	The schematic representation of a Encoder-Decoder RNN structure. . . . .	26
3.16	The two strategies to combine CNN and RNN. . . . .	28
3.17	The grid-search results of hyperparameter tuning of the CNNLSTM model . .	29
3.18	Example of GPS constellation (U.S. Coast Guard Navigation Center, 2021) . .	31

4.1	The RMS w.r.t. time of the five ML models and the original ultra-rapid products. The four sub-figures show the Radial RMS, Tangential RMS, Normal RMS, 3D RMS, respectively. . . . .	34
4.2	The relative improvements of individual samples using the five ML models w.r.t. the original ultra-rapid products in percentage. The four sub-figures show the Radial improvements, Tangential improvements, Normal improvements, 3D improvements, respectively. . . . .	36
4.3	Two selected samples with (down) and without strong error accumulation (up) and the corresponding prediction results using LSTM. . . . .	41
4.4	The $RMS_t$ of the four basic DL models and the original ultra-rapid products. The four sub-figures show the a) Radial $RMS_t$ , b) Tangential $RMS_t$ , c) Normal $RMS_t$ , d) 3D $RMS_t$ , respectively. . . . .	43
4.5	The relative improvements of $RMS_i$ using the four basic DL models w.r.t. the original ultra-rapid products in percentage. The four sub-figures show the a) Radial improvements, b) Tangential improvements, c) Normal improvements, d) 3D improvements, respectively. . . . .	44
4.6	The $RMS_t$ of the five different LSTM structures and the original ultra-rapid products. The four sub-figures show the a) Radial $RMS_t$ , b) Tangential $RMS_t$ , c) Normal $RMS_t$ , d) 3D $RMS_t$ , respectively. . . . .	45
4.7	The relative improvements of $RMS_i$ using the five different LSTM structures w.r.t. the original ultra-rapid products in percentage. The four sub-figures show the a) Radial improvements, b) Tangential improvements, c) Normal improvements, d) 3D improvements, respectively. . . . .	48
4.8	The $RMS_t$ using LSTM with different features and the original ultra-rapid products. The four sub-figures show the a) Radial $RMS_t$ , b) Tangential $RMS_t$ , c) Normal $RMS_t$ , d) 3D $RMS_t$ , respectively. . . . .	50
4.9	The relative improvements of $RMS_i$ in the original experiment and the three experiments about solar activity. The four sub-figures show the a) Radial improvements, b) Tangential improvements, c) Normal improvements, d) 3D improvements, respectively. . . . .	51



4.10	The $\text{RMS}_t$ of the original orbits and improved orbits using LSTM of the GPS constellation (denoted by G) and GLONASS constellation (denoted by R). The four sub-figures show the a) Radial RMS, b) Tangential RMS, c) Normal RMS, d) 3D RMS, respectively. . . . .	52
4.11	The relative improvements of $\text{RMS}_i$ of the GLONASS and GPS constellations using LSTM. The four sub-figures show the a) Radial improvements, b) Tangential improvements, c) Normal improvements, d) 3D improvements, respectively.	53
A.1	The version with absolute values in centimeter corresponds to Fig. 4.2 . . . . .	56
A.2	The version with absolute values in centimeter corresponds to Fig. 4.11 . . . . .	57
A.3	The version with absolute values in centimeter corresponds to Fig. 4.5 . . . . .	57
A.4	The version with absolute values in centimeter corresponds to Fig. 4.7 . . . . .	58
A.5	The version with absolute values in centimeter corresponds to Fig. 4.9 . . . . .	58

# List of Tables

2.1	IGS products . . . . .	5
3.1	Structure of the DNN model in this study. . . . .	18
3.2	Structure of the CNN model in this study. . . . .	20
3.3	Structure of the LSTM model in this study. . . . .	23
3.4	Structure of the GRU model in this study. . . . .	25
3.5	Structure of the biLSTM model in this study. . . . .	26
3.6	Structure of the EDLSTM model in this study. . . . .	27
3.7	Structure of the CNNLSTM model in this study. . . . .	29
4.1	The maximal, averaged, minimal absolute improvements w.r.t. time in centimeter of the five models. The best performances are highlighted. . . . .	35
4.2	The positive ratio of the five ML models [%]. . . . .	35
4.3	The maximal, averaged, minimal absolute improvements w.r.t. time in centimeter of MLP and MLP with inclusion of solar parameters. The best performances are highlighted. . . . .	40
4.4	The maximal, averaged, minimal absolute improvements of $RMS_t$ in centimeter of the four models. The best performances are highlighted. . . . .	43
4.5	The positive ratio of the four basic DL models [%]. . . . .	44
4.6	The maximal, averaged, minimal absolute improvements of $RMS_t$ in centimeter of the five different LSTM structures. The best performances are highlighted. . . . .	46
4.7	The positive ratio using the five different LSTM variants [%]. . . . .	47

4.8	The maximal, averaged, minimal absolute improvements of $\text{RMS}_t$ in centimeter of the three experiments and the original experiment using LSTM. The best performances are highlighted. . . . .	49
4.9	The positive ratio of the four experiments [%]. . . . .	50
4.10	The maximal, averaged, minimal absolute improvements w.r.t. time in centimeter of GLONASS and GPS using LSTM. The best performances are highlighted.	52
4.11	The positive ratio of the improved GLONASS and GPS orbits using LSTM [%].	53

# Chapter 1

## Introduction

The amount of space objects around the Earth has increased rapidly in recent years, which causes increasingly high risks of collision between space objects. Therefore, precise orbit determination (POD) and prediction are vital for space science. More specifically, GNSS satellites are some of the most important satellites for the geodetic community, and the accurate prediction of their orbits is essential for many real-time geodetic applications, including real-time navigation. The typical way to obtain accurate orbit predictions is using physics-based orbit propagators. However, the success of physics-based propagators requires precise initial states of the space object and good knowledge of the environmental information, which are the two significant difficulties for POD. The insufficient modeling of the changing perturbing forces also causes error accumulation with time, which is one of the major remaining challenges of POD.

Motivated by the rapid expansion of computing power and the considerable data volume in recent years, more and more scientists have applied machine learning (ML) and deep learning (DL) in geoscience (Bergen et al., 2019; Reichstein et al., 2019), geophysics (Yu and Ma, 2021) and geodesy (Butt et al., 2021). More specifically, in the community of space geodesy, many applications of ML and DL have also shown promising results, such as predicting the time series of the Earth orientation parameters (Schuh et al., 2002; Gou et al., 2021), super resolving space geodetic data (Seyoum et al., 2019; Irrgang et al., 2020) and modelling the time series of GNSS station coordinates (Shahvandi and Soja, 2021; Crocetti et al., 2021; Ruttner et al., 2022). Inspired by these successes and the time series characteristics of the satellite orbits, we can apply ML and DL algorithms to assess if they can be used to further reduce orbit errors.

There are only a few studies that try to improve POD using ML or DL approaches. Pihlajasalo et al. (2018) investigated the potential of using Convolutional Neural Networks (CNN) to improve the accuracy of their orbit prediction model. They computed the orbit errors based on the broadcast ephemerides provided by IGS and converted them into an image-like format to feed into a very simple CNN model. The improvements on all orbit types of GPS and Beidou constellations have been achieved. Besides, more studies focus on the Low Earth Orbit (LEO) satellites because the influences of non-gravitational perturbing forces are more significant than other types of orbits. Peng and Bai (2018*b*) have tested the performance of another ML model called Support Vector Machine (SVM) on the LEO satellites in a simulation-based environment. Later, they applied the same strategy on publicly available Two-Line Element (TLE). In these two studies, they conclude that SVM can significantly improve the accuracy on both position and velocity when the data volume is big enough. Mortlock and Kassas (2021) built up a Time Delay Neural Network (TDNN) for LEO satellites, which reduces errors in tracking LEO satellite positions and effectively improves a vehicle's navigation performance. However, all the mentioned studies model the satellite orbits with the accuracy of dozens of meters or even kilometers. None of them study the feasibility of ML and DL algorithms for improving the POD at meter or centimeter level.

In this study, we focus on the orbit prediction of GNSS constellations and challenge centimeter-level orbits. We investigate the potential of using different ML and DL algorithms for improving the accuracy of the ultra-rapid products from IGS. As ground truth we consider the IGS final products, and the differences between the ultra-rapid and final products are computed and serve as targets for the ML and DL methods. In this context, we combine the advantages of physics-based and data-driven ML and DL methods. To further study the impacts of insufficient modelling of non-gravitation forces, we also consider different related parameters as additional features to implicitly model the solar impact since the solar radiation pressure forces are supposed to be most significant on the altitudes of GNSS orbits (Montenbruck and Gill, 2012). The geometric relations between the Sun, the satellite, and the Earth are also considered to model the effect of solar radiation more accurately. Besides, since we have investigated a huge amount of different ML and DL algorithms, an emphasis of this study is the detailed discussion about the advantages and disadvantages of different algorithms for the application of POD.

The rest of this report is organized as follows. In Chapter 2, the used data are described. In Chapter 3, we explain the used data preprocessing strategy followed by the description of used ML and DL algorithms. The results and discussions are reported in Chapter 4. Finally, we conclude our study and propose some potential further work in Chapter 5.

# Chapter 2

## Data

In this section, the data we used in this study is introduced. Firstly, the GNSS data are treated, followed by the solar data.

### 2.1 GNSS Data

To analyse the satellite orbit predictions, GNSS data is used. Different data providers such as the U.S. GPS, the European Galileo, the Chinese BeiDou, and the Russian GLONASS regularly offer accurate orbit positions of their satellite constellations. For this project, GPS data was used. The GPS constellation consists of 32 active satellites orbiting on six planes (Rothacher, 2021). The GFZ Data Service, which acts as an analysis center of the International GNSS Service (IGS), provides different orbit products, including broadcast, ultra-rapid, rapid, and final orbit data (IGS, 2021). Table 2.1 shows the comparison of these orbit products in accuracy and latency.

The final orbit serves as a reference for the orbit data (Männel et al., 2020*b,a*). The differences within the 24-hour observations interval of the ultra-rapid products are used as input features, whereas the differences within the 24-hour prediction interval of the same ultra-rapid products serve as targets. The ultra-rapid products (observation and prediction) are available every 6 hours and consist of 24 hours observation data and 24 hours prediction data. The availability of the ultra-rapid products is visualized in figure 2.1. The final orbit products are available with a two-week delay, as shown in Table 2.1. The position data of

Table 2.1: IGS products

Type	Accuracy	Latency
Broadcast	100cm	real time
Ultra-rapid prediction	5cm	real time
Ultra-rapid observation	3cm	6 hours
Rapid	2.5cm	1 day
Final	2.5cm	2 weeks

ultra-rapid as well as final orbits are available at 15-minute intervals. In this study, GPS data from January 2019 to April 2021 is used, which includes over 200'000 data samples.

As required for machine learning tasks, the data samples are divided into training, validation and test data with proportions of 70% (144'075) and 15%(30'873) each. In a further step of this project, GLONASS orbit data is included for additional analysis. This data are also provided by IGS (IGS, 2021).



Figure 2.1: Schematic diagram of the IGS Ultra-rapid orbit products provided by GFZ

## 2.2 Solar Data

The movement of the satellites orbiting around the earth is influenced by solar activity. The effect of solar radiation pressure varies, among other things, with the occurring number and size of sunspots, causing an almost periodic solar cycle (NASA Space Place, 2021). To analyse the influence of solar activity on the satellite orbits, several solar parameters are incorporated in this study. These are the solar flux, the  $K_p$  index, and the  $C_{0,0}$  term of the ionosphere. Additionally, solar positions are included to enable a projection of the value of solar activity onto the three dimensions in space.



### 2.2.1 Solar Flux

One parameter, which describes solar activity is solar flux. The solar flux corresponds to the spectral flux density of solar radio radiation at the frequency of 2800 MHz. This corresponds to a wavelength of 10.7cm, which is why the term *10.7cm Solar Flux* is used. The 10.7cm Solar Flux is proportional to solar activity and correlates with the number of sunspots. In this study, data provided by Space Weather Canada is used. They provide three different values of solar flux, the *observed*, the *adjusted*, and the *Series D* flux. In this case, the *observed* values are used. They are measured by solar radio telescopes and used when dealing with terrestrial observations. (Space Weather Canada, 2021)

The 10.7cm Solar Flux values are measured three times per day. The data describing solar activity is needed in the same 15-minute interval as the GNSS data because they are incorporated as additional features in the process of Machine Learning. A linear interpolation is therefore carried out based on the median values formed from the three daily solar flux values of the same period as the GNSS data, from January 2019 to April 2021.

### 2.2.2 $K_p$ index

Another parameter, which is included to describe solar activity is the  $K_p$  index. The  $K_p$  index was developed by Julius Bartels in 1949 as a measure for geomagnetic activity. It describes solar particle radiation via its magnetic effects, represented by a value between 0 and 9.  $K_p$  index values from Matzka et al. (2021) are used in this study. They provide data in a time interval of three hours. As described in the section above, the data samples are needed in a 15-minute interval. A linear interpolation was therefore also carried out with the  $KP$ -values of the period from January 2019 to April 2021.

### 2.2.3 $C_{0,0}$ term of ionosphere

The Total Electron Content (TEC) is the number of electrons along a path through the ionosphere. Depending on the sunspot number, the TEC varies significantly and is therefore a suitable measure to describe solar activity (Space Weather Prediction Center, 2021). In this study, the TEC values, or  $C_{0,0}$  terms of ionosphere, provided by the University of Bern are used. The values are monitored hourly and are expressed in total electron content units

(TECU) (University of Bern, 2021). In this case, too, the values are required at 15-minute intervals. The missing values are completed with linear interpolation.

### 2.2.4 Solar Position

To include the position of the sun, beta angles and ephemerides of the sun are incorporated. The beta angle describes the angle between the vector pointing from the earth towards the sun and the projection of this vector onto the orbital plane. The value of the beta angle varies between  $+75^\circ$  and  $-75^\circ$  with an annual period (Alpha Magnetic Spectrometer, 2021). Beta angles are calculated based on the orbital elements of the satellites as well as the ecliptic true solar longitude and the equatorial sun elevation. In this study, programs provided by the supervisors are used to compute the beta angles for the required period from January 2019 to April 2021. Data of the ecliptic true solar longitude is used from *Astro Pixels* (Fred Espenak, 2021).

# Chapter 3

## Methodology

In this chapter, the methodology of this study is introduced. This includes an overview of the process, an explanation of the data preprocessing, and descriptions of the used algorithms and the experiments on feature inclusion and data grouping.

### 3.1 Overview

Figure 3.1 shows an overview of the pipeline of this study. Firstly, the orbit errors between the ultra-rapid products and final products are computed. Since the original products are in the Earth-centered Earth-fixed frame (ECEF), we should transform the orbit errors into Earth-centered initial (ECI) and Radial-Tangential-Normal (RTN) frame, which is a satellite-fixed frame. The R, T, N differences between the ultra-rapid and final products within the 24-hours observation interval serve as basic input features for the analysis. The targets of the ML and DL models are the differences within the 24-hours prediction interval of the same ultra-rapid product. In addition, solar positions and solar activity parameters are included for additional experiments. Since this approach aims to predict the orbit errors, we are dealing with a regression problem. Multiple ML and DL algorithms for supervised regression are therefore applied to the orbit differences. To gain the best performance, we should three different models for the three targets.

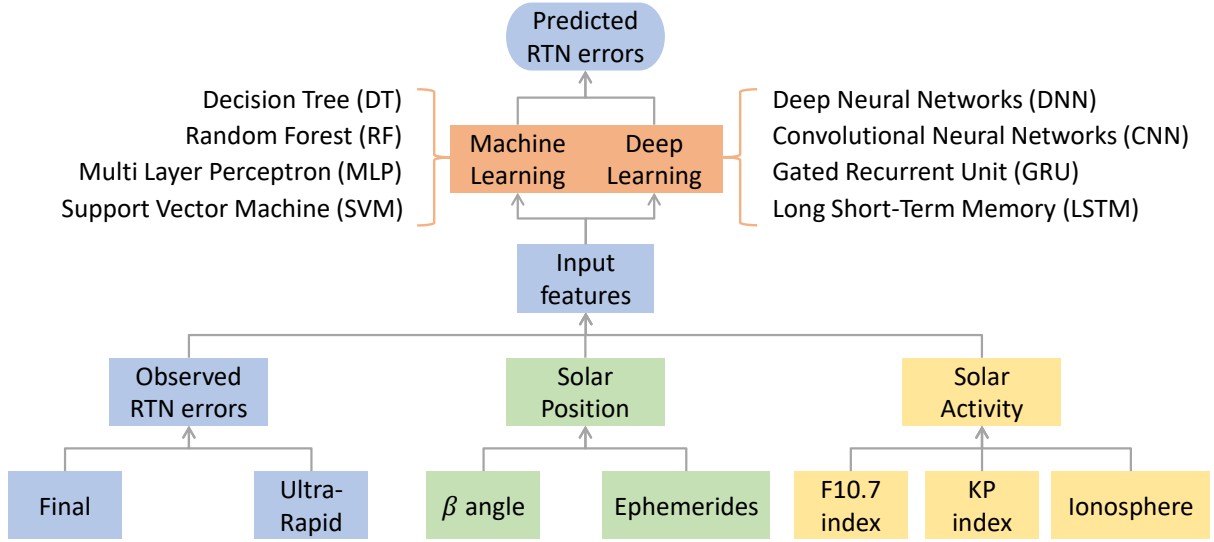


Figure 3.1: Graphical representation of the pipeline of the study

## 3.2 Data Preprocessing

### 3.2.1 Generate Orbit Errors

Before we apply the ML and DL algorithms, we should preprocess the original products. We should first align each ultra-rapid product to the corresponding final product based on the GPS time (GPST). Then, the ECEF coordinates should be transformed into the ECI frame for further processing. The connections between the ECEF and ECI are the Earth orientation parameters (EOPs). Therefore, we can realize the transformation using the following equation:

$$\mathbf{r}_i = \mathbf{PNUXY} \cdot \mathbf{r}_e = \mathbf{R}_e^i \cdot \mathbf{r}_e \quad (3.1)$$

where  $\mathbf{r}_i$  and  $\mathbf{r}_e$  denote the orbit vector in the ECI and ECEF frames, respectively.  $\mathbf{P}$  and  $\mathbf{N}$  are the precession and nutation matrices, respectively.  $\mathbf{U}$  denotes the rotation matrix, which is related to dUT1.  $\mathbf{X}$  and  $\mathbf{Y}$  are the rotation matrices describing the polar motion. All the EOPs can be found from IERS. Besides, IERS also provides the Earth rotation matrix  $\mathbf{R}_e^i$  including the full set of EOPs and diurnal and semi-diurnal variations produced by ocean tides in polar motion and UT1 (Petit and Luzum, 2010). For convenience, we directly downloaded the Earth rotation matrix and applied them to our orbit data.

Once we have the satellite orbits in the ECI system, we should first compute the orbit error between the ultra-rapid orbit ( $\mathbf{r}_{i,\text{Final}}$ ) and the final orbit ( $\mathbf{r}_{i,\text{UR}}$ ):

$$\Delta \mathbf{r}_i = \mathbf{r}_{i,\text{UR}} - \mathbf{r}_{i,\text{Final}} \quad (3.2)$$

Then, we should transform the orbit errors into the RTN frame:

$$\mathbf{e}_R = \frac{\mathbf{r}_i}{\|\mathbf{r}_i\|} \quad (3.3)$$

$$\mathbf{e}_T = \frac{\mathbf{r}_i \times \mathbf{v}_i}{\|\mathbf{r}_i \times \mathbf{v}_i\|} \quad (3.4)$$

$$\mathbf{e}_N = \mathbf{e}_T \times \mathbf{e}_R \quad (3.5)$$

$$\mathbf{R}_i^s = \begin{bmatrix} \mathbf{e}_R & \mathbf{e}_T & \mathbf{e}_N \end{bmatrix}^T \quad (3.6)$$

$$\Delta \mathbf{r}_s = \mathbf{R}_i^s \cdot \Delta \mathbf{r}_i \quad (3.7)$$

where  $\mathbf{e}_R$ ,  $\mathbf{e}_T$ ,  $\mathbf{e}_N$  are the three normal vectors in the RTN frame.  $\mathbf{R}_i^s$  denotes the transformation matrix.  $\Delta \mathbf{r}_s$  denotes the error vector in the RTN frame. From Eq. (3.3) to Eq. (3.7) we can find that the velocity in the ECI frame is necessary. Since the original products do not provide any information about the velocities, we have to compute them using numerical differential:

$$\mathbf{v}_{i,t} = \frac{\mathbf{r}_{i,t+1} - \mathbf{r}_{i,t-1}}{2 \cdot \delta t} \quad (3.8)$$

where  $\delta t$  is the time difference between two adjacent epochs. In our case is 15 min. However, we can not have any information about the velocity at the very beginning and end of each ultra-rapid product as indicated in Eq. (3.8), which means we can only obtain 190 valid data from 192 epochs.

### 3.2.2 Feature Standardization

Since well-standardized features can simplify the optimization process in the ML and DL algorithms and therefore improve their performances (Goodfellow et al., 2016), we should also standardize our input features. Since our data are time series and also accumulate with time, as shown in Fig. 3.2, we should not compute the mean values and standard deviations of each feature, but compute the mean values and standard deviations of each feature at each individual time epoch, namely:

$$\hat{f}_{i,t} = \frac{f_{i,t} - \mu_{i,t}}{\sigma_{i,t}} \quad (3.9)$$

where  $f_{i,t}$  and  $\hat{f}_{i,t}$  denote the original and standardized  $i^{\text{th}}$  feature at the time epoch  $t$ , respectively.  $\mu_{i,t}$  and  $\sigma_{i,t}$  are the mean value and standard deviation of the  $i^{\text{th}}$  feature at the epoch  $t$ . We use the *StandardScaler* in *Scikit-learn* (Pedregosa et al., 2011) to realize the standardization. We first train 95 scalers for each feature and also 95 scalers for the target based on the training set. Later, the scalers will be applied on the validation set and test set.

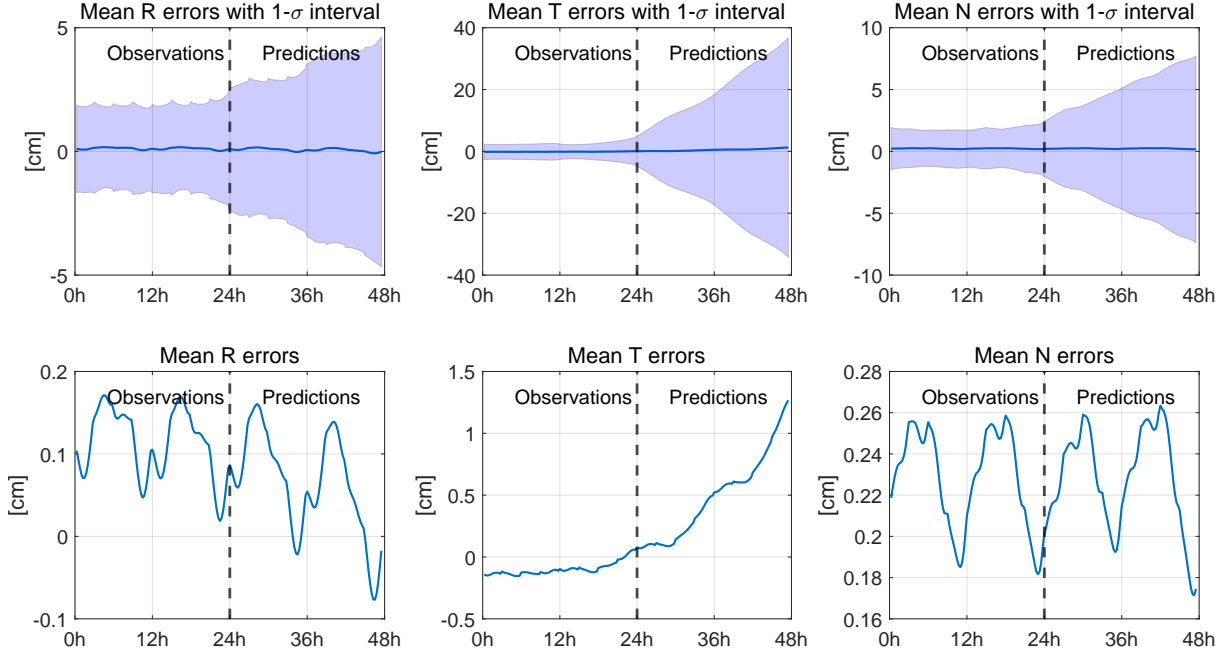


Figure 3.2: The mean orbit errors of the GPS constellations with  $1-\sigma$  interval (up) and the zoom-in version of the mean orbit errors (down).

This standardization strategy is applied to all the input features (R, T, N errors as well as additional features). Fig. 3.3 shows thirty examples of the RTN errors and corresponding standardized RTN errors. The first column shows the radial errors, where we can find that the proposed standardization can clearly reduce the magnitude difference and keep the major features at the same time. The blue, yellow, and orange lines still have significantly larger magnitudes than others with similar patterns as the original errors. The advantage of the standardization is more significant for the tangential errors, where we can find that the severe accumulation problem is resolved.

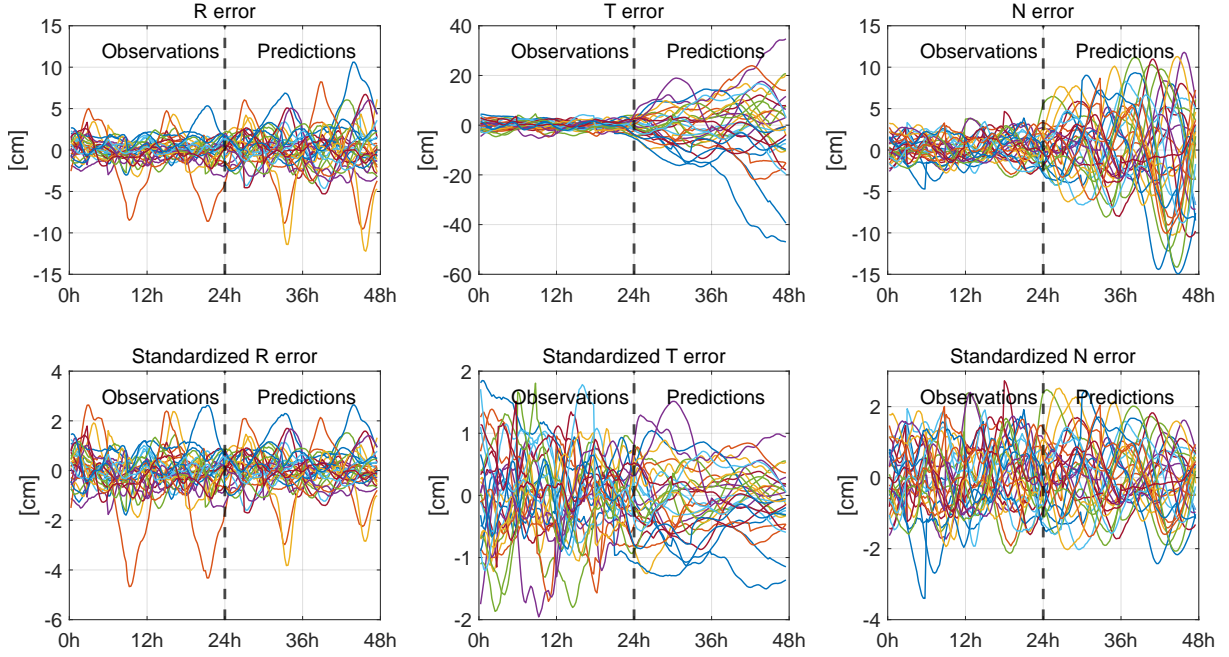


Figure 3.3: Selected individual samples of R, T, N errors (up) and corresponding standardized R, T, N errors (down) of the GPS constellation.

### 3.3 Machine Learning Algorithms

In this Chapter, the machine learning algorithms, which are applied in this study, are introduced. Five different algorithms have been tested, namely *Linear Regression*, *Decision Tree*, *Random Forest*, *Multi Layer Perceptron*, and *Support Vector Machine*. The free software library scikit-learn by Pedregosa et al. (2011) provides machine learning algorithms in the programming language python for the mentioned methods, which were used in this study.

In order to find the optimal hyperparameters for the respective algorithm, parameter tuning is carried out. There are different approaches to find the parameters which show the performance without over- or underfitting the model. Search functions like *Grid Search*, *Random Search* and *Bayesian Search* algorithms are provided by scikit-learn (Pedregosa et al., 2011). In this study, *Bayesian Search* is applied for the hyperparameter tuning. This method is less time-consuming than the other two and provides better results. In contrast to *Grid Search* and *Random Search*, *Bayesian Search* uses information from previous evaluations. This search algorithm is thus able to focus on the promising hyperparameters and does not necessarily cover the entire search field, which is the case for the other two. (Koehrsen, 2018)

### 3.3.1 Linear Regression

The simplest algorithm for regression is linear regression. The goal is to minimize the sum of the squared errors, which can be represented mathematically in the following way:

$$\min \sum_{i=1}^n (y_i - w_i x_i)^2$$

One can see that the squared difference between the target  $y_i$  and the sum of the coefficient  $w_i$  and the predictor  $x_i$  is minimized. (Sharp, Tom, 2021) The goal of linear regression is to find the best fitting line, which models the data points the best. The concept of the algorithm is shown in figure 3.4. (Goodfellow et al., 2016)

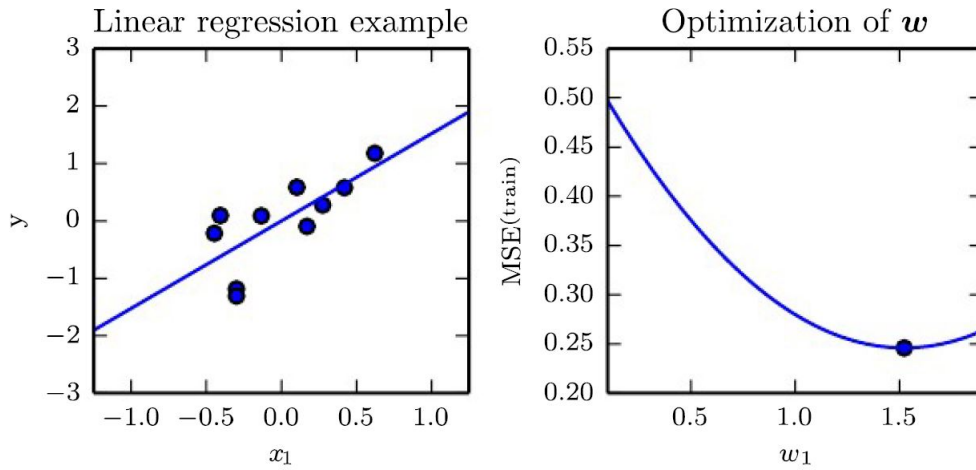


Figure 3.4: Concept of Linear Regression (Goodfellow et al., 2016)

In the left part of the figure, the best fitting line for the given data set is displayed and the right part shows the optimization of the parameters  $\omega$ . One can thus see that linear regression is dealing with an optimization problem. The linear regression algorithm provided by scikit-learn (Pedregosa et al., 2011) uses a number of parameters and attributes. Some of the important parameters to mention are *fit\_intercept*, which defines if the intercept is calculated for the model or not, and *normalize*, deciding if the regressor is normalized or not before regression (Pedregosa et al., 2011). As mentioned before, the best-performing hyperparameters are evaluated by Bayesian Search. Since linear regression is a simple algorithm for machine learning, the prediction results are not expected to be outstanding.



### 3.3.2 Decision Tree

Decision Tree (DT) is a common supervised machine learning model which is used for classification and regression tasks. The trees consist of nodes, branches, and leaves. A simple visualization of a decision tree architecture is shown in figure 3.5. The target at the root node is split into targets at the sub-nodes, which are pursued where the weighted mean square error is the lowest. This process goes on until the dataset is split into its smallest parts. In the case of regression, the targets are real numbers. (Jena and Dehuri, 2020)

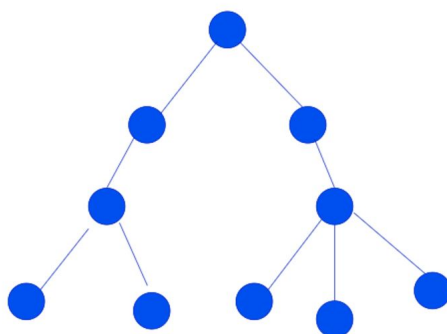


Figure 3.5: Visualization of Decision Tree model (Das, Nipa, 2020)

Scikit-Learn also provides a predefined function for this machine learning model. Some of the most important parameters of the decision tree function provided by scikit-learn are: *max\_depth*, *min\_samples\_split*, *min\_samples\_leaf*, *max\_features*. *max\_depth* stands for the maximum depth of the tree, *min\_samples\_split* is the minimal value of samples to split a node, *min\_samples\_leaf* is the minimal value of samples to be at a leaf node, and *max\_features* is the number of features to include for the best split. (Pedregosa et al., 2011) It is crucial to find the appropriated values for these parameters to avoid over- and underfitting. As has already been pointed out, Bayesian Search is used to find the optimal parameters. Decision tree is a quite powerful machine learning model and we thus expect improvements through the application of this method.

### 3.3.3 Random Forest

To obtain more accurate results than a single model can deliver, ensemble models are used. Random forest (RF) is a widely used supervised machine learning model which is composed

of multiple decision trees. A simple visualization of the concept of random forest is shown in figure 3.6.

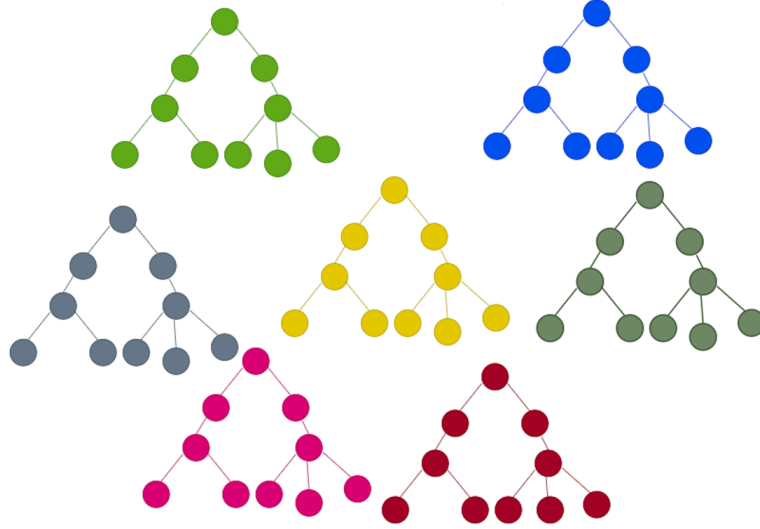


Figure 3.6: Visualization of Random Forest model (Das, Nipa, 2020)

The data is randomly split into subsets which are then processed separately in decision trees and are therefore less correlated. The final results are then obtained by averaging the results of the individual trees. (Vladimir Lyashenko, 2021) The parameters that scikit-learn uses for the implementation of a random forest model are among others: *n\_estimators*, *max\_depth*, *min\_samples\_split*, *min\_samples\_leaf*, *max\_features*. The number of trees in the forest is given by *n\_estimators*. The other parameters are similar to the ones used for decision tree and characterize the individual trees. (Pedregosa et al., 2011) Also for this model, Bayesian Search is used in this study to find the best performing hyperparameters. Random forest is a powerful and robust model and is thus expected to show improvements that are better than the ones delivered from linear regression and decision tree.

### 3.3.4 Support Vector Machine

The fourth machine learning algorithm that is applied to the orbit data is support vector machine (SVM). The goal of this algorithm is to find the best fitting line which represents the trend of the dataset, this line is called *hyperplane*. In contrast to a simple linear regression, support vector machine uses a constraint that defines the maximum error  $\varepsilon$  that is accepted.

The objective function and the corresponding constraint are mathematically represented in the following way:

$$\min \frac{1}{2} \|w\|^2$$

$$|y_i - w_i x_i| \leq \varepsilon$$

While linear regression minimizes the sum of the squared errors, support vector machine minimizes the  $l_2$  norm of the coefficient  $w$ . The error margins (defined by the constraint) form so-called *decision boundaries* around the hyperplane. Points lying in the range between the decision boundaries are taken into account. Figure 3.7 shows a visualization of the support vector machine model.

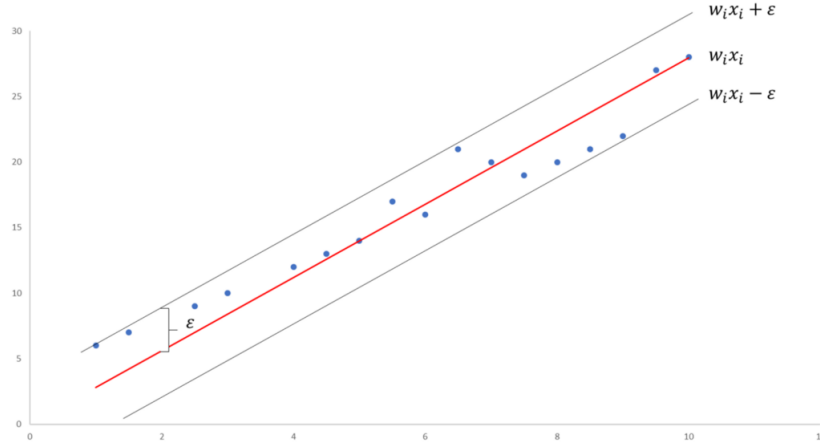


Figure 3.7: Visualization of Support Vector Machine model (Sharp, Tom, 2021)

The red line represents the hyperplane, the two black lines are the decision boundaries at distance  $\varepsilon$  from the hyperplane. (Suthaharan, 2016; Sharp, Tom, 2021) Relevant parameters of the support vector machine algorithm provided by scikit-learn are: *epsilon*, *kernel*, *max\_iter*. The maximum accepted error is represented by *epsilon*, *kernel* describes the kernel type (linear, polynomial, etc) which is used and *max\_iter* is the number of iterations computed by the solver. (Pedregosa et al., 2011) To find the optimal values of these hyperparameters, bayesian search is applied.

### 3.3.5 Multi Layer Perceptron

The model of multi layer perceptron (MLP) belongs to the class of feed-forward artificial neural networks and is, therefore, a supervised deep learning model. However, in this study, the model of multi layer perceptron is assigned to the Chapter of machine learning algorithms. In the Chapter on deep learning algorithms, more advanced deep neural networks like convolutional and recurrent neural networks are included. A MLP is composed of input layers, hidden layers, and output layers. A simple visualization of such a model is shown in Figure 3.8.

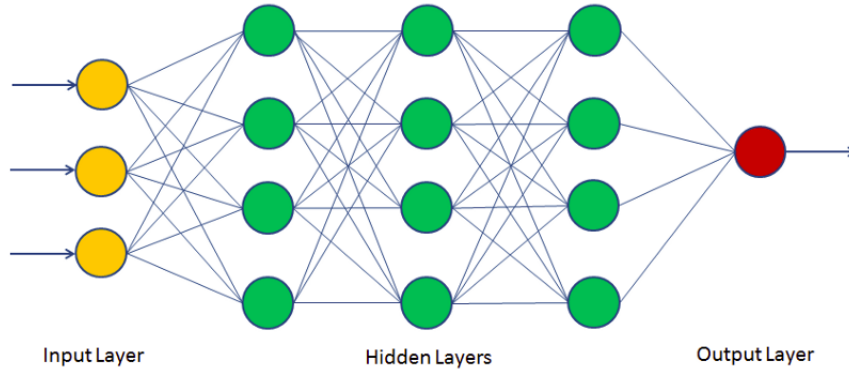


Figure 3.8: Visualization of Multi Layer Perceptron model (Avinash Navlani, 2020)

As the name *feed forward* artificial neural network indicates, data in a MLP flows in a forward direction from input to output layer. The hidden layers consist of neurons with activation functions, which define the output based on the input. The number of hidden layers and the number of neurons contained in each layer can be arbitrarily chosen. The results (weighted sums) of each layer are passed to the next one. During this process, the model is able to learn due to the backpropagation function. The output (gradient of mean squared error) of the neural network is compared with the desired output. The weights are then adjusted accordingly in the backward direction from the output layer to the input layer. (Elliott, Stephen, 2000; Bento, Carolina, 2021)

The predefined function of a MLP provided by scikit-learn is used in this study. The most crucial parameters to set in this case are among others: *hidden\_layer\_sizes*, *activation*, *solver*, *max\_iter*. The *hidden\_layer\_sizes* stands for the size of the hidden layers contained in the neural network. For the sake of simplicity, we chose three hidden layers for this study and determined the best performing sizes of the three layers. *activation* defines the activation function, where there is a choice of four functions. The parameter *solver* stands for the solver

of weight optimization with three optimizers to choose from. *max\_iter* is the number of iterations carried out by the solver. (Pedregosa et al., 2011) Bayesian Search is also used for the hyperparameter tuning of this model.

## 3.4 Deep Learning Algorithms

In this section, we will introduce the basic principles of the deep learning methods that we used in this study. Three major categories of neural networks will be introduced, namely Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN). In addition, different variations of RNN will be introduced in detail, including different neurons and different structures of the networks. All the realizations of the models are based on *TensorFlow V2.4.0* (Abadi et al., 2015).

### 3.4.1 Deep Neural Network

The first model is a simple DNN model, which is similar to the MLP model introduced in Section 3.3.5. The difference between DNN and MLP in this study is the number of layers. We design a DNN with eight hidden layers with *tanh* as the activation function followed by one output layer, as shown in Table 3.1.

Table 3.1: Structure of the DNN model in this study.

Model	Layer	Activation	Number of neurons	Number of parameters
DNN	Dense	tanh	128	36'608
	Dense	tanh	128	16'512
	Dense	tanh	64	8'256
	Dense	tanh	64	4'160
	Dense	tanh	32	2'080
	Dense	tanh	32	1'056
	Dense	tanh	16	528
	Dense	tanh	16	272
	Dense	-	95	1'615

### 3.4.2 Convolutional Neural Network

The CNN (LeCun et al., 1998) is a specific type of neural networks that uses convolution in place of general matrix multiplication in some layers. The most successful applications of CNN are in the field of computer vision, where CNN-based models are used to process image data. Moreover, CNN-based models can also be applied to time series data using the 1D convolutional operation. A more specific model called Temporal Convolutional Networks (TCN) is introduced by Lea et al. (2016). TCN uses the causal convolution, meaning that the element at epoch  $t$  will only be convoluted with the non-latter elements to ensure the information in the future will not leak back to influence the past hidden state or output, as shown in Fig. 3.9.

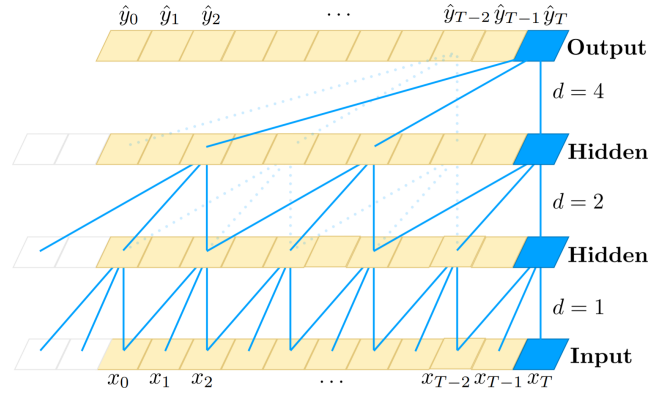


Figure 3.9: The schematic representation of a TCN model (Bai et al., 2018).

In our case, however, the causal convolution is unnecessary since our inputs are in the 24h observation interval, whereas the outputs are in the 24-hours prediction interval. We can use all the information within the observation interval to predict our target in the prediction interval. We designed a CNN model as shown in Fig. 3.10. Considering the temporal resolution of 15 min, we use the kernel size of four, meaning that each neuron in the first hidden layer contains information of 1 h. Besides, we use the hyperbolic tangent activation function ( $\tanh$ ) instead of  $ReLU$  since the negative values are also crucial for the time series prediction. As a result, we cannot apply max-pooling layers in our model. The influences of different activation functions and pooling layers have been investigated during this study. The results fit our expectation that the model using  $\tanh$  as the activation function and without including pooling layers give the best performance. After two 1D convolutional layers, we use a flattening layer to convert the 2D features into a 1D vector. Then, a fully connected layer with 64 hidden

neurons and activation function  $\tanh$  is applied to learn the non-linearity further. In the end, an output layer is applied. The whole structure and number of trainable parameters are summarized in Table 3.2.

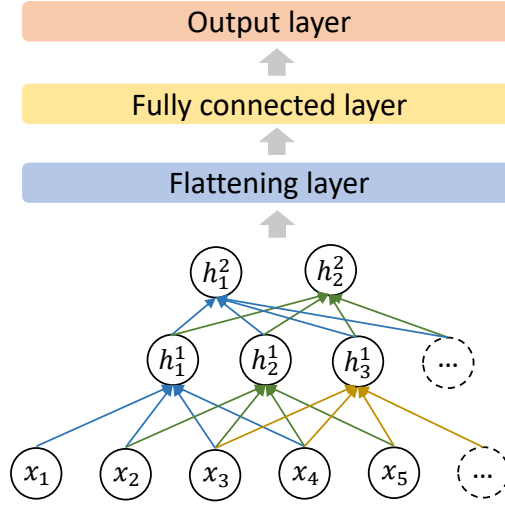


Figure 3.10: The schematic representation of the used CNN model in this study.

Table 3.2: Structure of the CNN model in this study.

Model	Layer	Activation	Output shape	Number of parameters
CNN	Conv1D	$\tanh$	(None, 92, 16)	208
	Conv1D	$\tanh$	(None, 89, 16)	1'040
	Flatten	-	(None, 1424)	0
	Dense	$\tanh$	(None, 64)	91'200
	Dense	-	(None, 95)	6'175

### 3.4.3 Recurrent Neural Network

The biggest shortcoming of the traditional neural networks is that they cannot remember the sequential information. As a result, the performance of the traditional NNs in analyzing a long time series is limited. One of the most preferred DL models for processing sequential

data is called RNN (Rumelhart et al., 1986). The key idea of RNN is that the hidden state at epoch  $t$  is related to the external signals ( $\mathbf{x}_t$ ) and previous state  $\mathbf{h}_{t-1}$ , namely:

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t; \theta) \quad (3.10)$$

If we unfold Eq. (3.10) we can get:

$$\mathbf{h}_t = g(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t; \theta) \quad (3.11)$$

namely the hidden state at epoch  $t$  depends on all the previous inputs. This unfolding process introduces two major advantages. First, the input size of the learned model is independent of the length of the input sequence. Second, different epochs share the same transition function  $f$ , which means they share the parameters (Goodfellow et al., 2016).

Based on the basic idea of RNN, many different structures have been developed. The most common structure is that at each epoch  $t$ , the model provides an output  $o_t$ , as shown in Fig. 3.11a. This kind of model can be used to do real-time monitoring. Another structure is shown in Fig. 3.11b. This model only has one output at the end of the sequence, which is usually used to summarize a sequence and give one fixed-size representation for further processing. The difference between these two structures and more advanced structures will be discussed in the rest of this section.

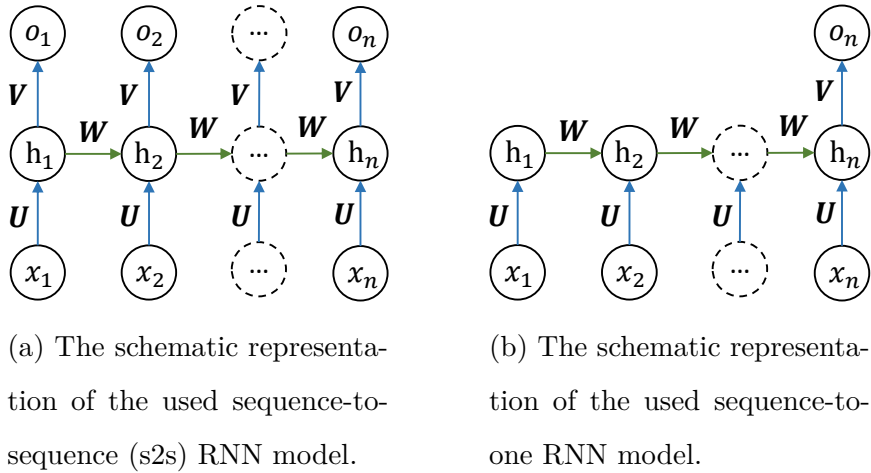


Figure 3.11: Comparison between the sequence-to-sequence structure and sequence-to-one structure of RNN.

Besides the standard backpropagation strategy that we use to train a normal NN, we have to apply a new strategy to train the RNN models back through time, called Back Propagation



Through Time (BPTT). When the sequence is long, we will face the problem of multiplying multiple gradients. This will cause major difficulties to train an RNN model, namely the gradient vanishing problem when the gradients are too small or the gradient exploding problem when the gradients are too big (Bengio et al., 1994; Pascanu et al., 2013). In the following sections, we will introduce two variants of the normal RNN neurons which can reduce this problem to a certain degree.

### 3.4.3.1 Long Short-Term Memory

The Long Short-Term Memory (LSTM) is one of the most famous RNN variants, which is first introduced by Hochreiter and Schmidhuber (1997). The LSTM neuron contains four gates, namely the input gate (**I**), forget gate (**F**), output gates (**O**), and cell state (**C**). Eq. (3.12) to Eq. (3.16) show a full set of mathematical representation of a LSTM neuron:

$$\mathbf{I}_{i,t} = \sigma \left( \mathbf{U}_i^I \cdot \mathbf{x}_t + \mathbf{W}_i^I \cdot \mathbf{h}_{t-1} + \mathbf{b}_i^I \right) \quad (3.12)$$

$$\mathbf{F}_{i,t} = \sigma \left( \mathbf{U}_i^F \cdot \mathbf{x}_t + \mathbf{W}_i^F \cdot \mathbf{h}_{t-1} + \mathbf{b}_i^F \right) \quad (3.13)$$

$$\mathbf{O}_{i,t} = \sigma \left( \mathbf{U}_i^O \cdot \mathbf{x}_t + \mathbf{W}_i^O \cdot \mathbf{h}_{t-1} + \mathbf{b}_i^O \right) \quad (3.14)$$

$$\mathbf{C}_{i,t} = \mathbf{F}_{i,t} * \mathbf{C}_{i,t-1} + \mathbf{I}_{i,t} * \sigma \left( \mathbf{U}_i^C \cdot \mathbf{x}_t + \mathbf{W}_i^C \cdot \mathbf{h}_{t-1} + \mathbf{b}_i^C \right) \quad (3.15)$$

$$\mathbf{h}_{i,t} = \tanh \left( \mathbf{C}_{i,t} \right) * \mathbf{O}_{i,t} \quad (3.16)$$

where  $\mathbf{U}$  and  $\mathbf{W}$  are trainable weight matrices whereas  $\mathbf{b}$  are biases. The subscripts denote the time epoch  $t$  of the  $i^{\text{th}}$  neuron within a layer.  $\sigma$  and  $\tanh$  denote the sigmoid and the hyperbolic tangent activation function, respectively.  $*$  is the element-wise product.

Fig. 3.12 shows a schematic representation of a LSTM neuron. The upper flow shows the transition of the cell state, whereas the under flow shows the transition of the hidden state through time. The first vertical gate is the forget gate, which controls whether the new state should contain the information from the previous state. If the second term of Eq. (3.15) close to zeros, the whole cell state will not be zeros due to the presence of the forget gate. The second vertical gate is the input gate, which imports the information from the input features  $\mathbf{x}_{i,t}$  and combines it with the previous hidden state. The last vertical gate is the output gate, which decides to which degree the next hidden state should be influenced.

The used LSTM model in this study is reported in Table 3.3, which follows the structure introduced in Fig. 3.11b. The output of the last LSTM layer will serve as the input of a fully

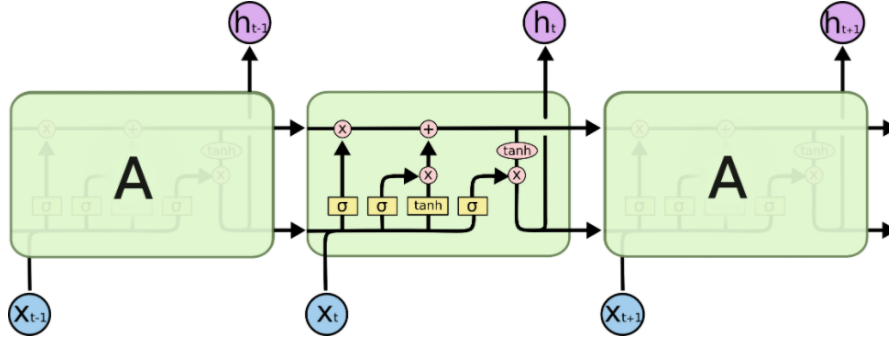


Figure 3.12: The schematic representation of a LSTM neuron (Colah, 2015)

connected layer with a hyperbolic tangent activation function. The last fully connected layer serves as the output layer and therefore without activation function. A certain limitation of this structure is that the final output will no longer be considered as a sequence by the network. As a comparison, another sequence-to-sequence LSTM model (Fig. 3.11a) is also built up. The results are shown in Section 4.2.2, which demonstrates the invalidity of the sequence-to-sequence structure for our purpose.

Table 3.3: Structure of the LSTM model in this study.

Model	Layer	Output shape	Number of parameters
	LSTM	(None, 95, 128)	67'584
	LSTM	(None, 95, 64)	49'408
LSTM	LSTM	(None, 32)	12'416
	Dense	(None, 128)	4'224
	Dense	(None, 95)	12'255

LSTM networks show a noticeable ability to learn long-term dependencies. However, the major disadvantage of such networks is that they have many trainable parameters, which means we need a huge dataset to properly train the model. To solve this problem, some variants and alternatives to LSTM are introduced.

### 3.4.3.2 Gated Recurrent Unit

Another famous RNN variant is Gated Recurrent Unit (GRU) introduced by Cho et al. (2014). As shown in Fig. 3.13, the GRU neuron consists of a reset gate ( $\mathbf{r}$ ), an update gate ( $\mathbf{z}$ ) and an

output gate where the candidate activation vector  $\tilde{\mathbf{h}}$  is generated. In this structure, the cell state and hidden state are merged. The mathematical representations are:

$$\mathbf{z}_{i,t} = \sigma(\mathbf{U}_i^z \cdot \mathbf{x}_t + \mathbf{W}_i^z \cdot \mathbf{h}_{t-1} + \mathbf{b}_i^z) \quad (3.17)$$

$$\mathbf{r}_{i,t} = \sigma(\mathbf{U}_i^r \cdot \mathbf{x}_t + \mathbf{W}_i^r \cdot \mathbf{h}_{t-1} + \mathbf{b}_i^r) \quad (3.18)$$

$$\tilde{\mathbf{h}}_{i,t} = \tanh(\mathbf{U}_i^h \cdot \mathbf{x}_t + \mathbf{W}_i^h (\mathbf{r}_{i,t} * \mathbf{h}_{t-1}) + \mathbf{b}_i^h) \quad (3.19)$$

$$\mathbf{h}_{i,t} = (1 - \mathbf{z}_{i,t}) * \mathbf{h}_{i,t-1} + \mathbf{z}_{i,t} * \tilde{\mathbf{h}}_{i,t} \quad (3.20)$$

where  $\mathbf{U}$  and  $\mathbf{W}$  are trainable weight matrices whereas  $\mathbf{b}$  are biases. The subscripts denote the time epoch  $t$  of the  $i^{\text{th}}$  neuron within a layer.  $\sigma$  and  $\tanh$  denote the sigmoid and the hyperbolic tangent activation function, respectively.  $*$  is the element-wise product.

We can immediately realize that the most significant advance of GRU compared to LSTM is that GRU contains less trainable parameters. This means that training a GRU network will be much easier than training the same network with LSTM neurons. On the other side, however, fewer parameters also mean less power in specific problems, especially with long time series.

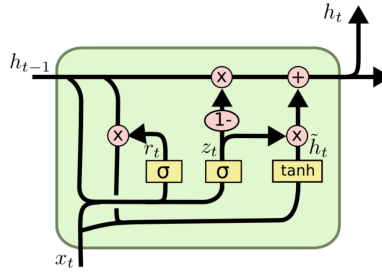


Figure 3.13: The schematic representation of a GRU neuron (Colah, 2015)

The used GRU model in this study is reported in Table 3.4. We keep the structures of the LSTM and GRU models the same to analyze the different performances of these two neurons in Section 4.2.1.

### 3.4.3.3 Bidirectional structure

As we mentioned above, the major problem of the basic sequence-to-sequence structure is that the output  $o_t$  does not contain any information of the latter epochs, such as  $x_{t+1}$ . This is true for real-time modelling and many time series prediction problems since we cannot get information from the future. However, in our case, the whole output sequence (prediction

Table 3.4: Structure of the GRU model in this study.

Model	Layer	Output shape	Number of parameters
	GRU	(None, 95, 128)	51'072
	GRU	(None, 95, 64)	37'248
GRU	GRU	(None, 32)	9'408
	Dense	(None, 128)	4'224
	Dense	(None, 95)	12'255

interval) is behind the whole input sequence (observation interval), which means every single output should contain all the information from the whole input sequence. One solution is called bidirectional RNN (biRNN), which is first introduced by Schuster and Paliwal (1997). The basic idea of biRNN is shown in Fig. 3.14 that it combines two RNNs. One moves forward through time, whereas the other moves backward through time. Therefore, the output  $o_t$  depends on both the future and the past, but more sensitive to the input values around time  $t$ .

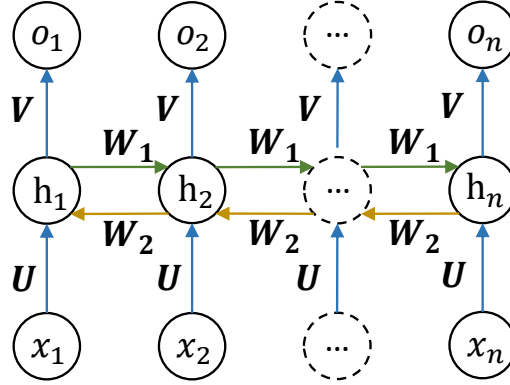


Figure 3.14: The schematic representation of a bidirectional RNN structure.

In our study, we combine the LSTM neuron with the biRNN structure to generate the bidirectional LSTM model (biLSTM) as shown in Table 3.5. The first *Dense* layer is followed by a *tanh* activation function, whereas the last one is without activation functions to serve as the output layer. We can clearly notice that one major shortcoming of this model is the number of parameters. Since the LSTM neuron is already complicated as described in Section 3.4.3.1 and the bidirectional structure further increases the complicity of the model, the number of

available training data might prevent successful training. The influence of the number of parameters will be discussed in detail in Section 4.2.2.

Table 3.5: Structure of the biLSTM model in this study.

Model	Layer	Output shape	Number of parameters
biLSTM	Bidirectional(LSTM)	(None, 95, 256)	135'168
	Bidirectional(LSTM)	(None, 95, 128)	164'352
	Bidirectional(LSTM)	(None, 95, 64)	41'216
	TimeDistributed(Dense)	(None, 95, 32)	2'080
	TimeDistributed(Dense)	(None, 95, 1)	33

#### 3.4.3.4 Encoder-Decoder structure

Another solution that allows us to include the information of the whole input sequence to predict the targets is called Encoder-Decoder structure (Cho et al., 2014; Sutskever et al., 2014). As shown in Fig. 3.15, the network consists of two parts. In the Encoder part, the input features  $\mathbf{x}_1$  to  $\mathbf{x}_n$  will be fed into an RNN model and generate a context vector  $\mathbf{C}$ , which represents a summary of the whole input sequence and serves as input features of the Decoder model. The context vector  $\mathbf{C}$  will be copied  $m$  times where  $m$  is the length of the output sequence. One of the most significant contributions of the Encoder-Decoder structure is that the output sequence does not have to have the same length as the input sequence. In our study, however,  $m = n = 95$ .

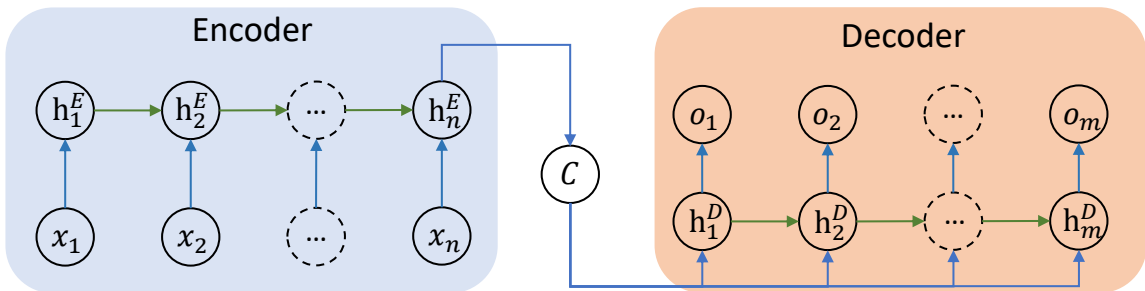


Figure 3.15: The schematic representation of a Encoder-Decoder RNN structure.

In this study, we also combine the LSTM neurons with the Encoder-Decoder structure to generate the model called Encoder-Decoder LSTM (EDLSTM), as shown in Table. Similar

to the biLSTM, the penultimate *Dense* layer is followed by an activation function, and the last one serves as the output layer. An explicit limitation of the EDLSTM is that the context vector cannot be too small. Otherwise, the Decoder cannot obtain sufficient information to predict the output sequence properly (Bahdanau et al., 2014). Therefore, we also face the problem of too many trainable parameters with this model, which will be discussed in detail in Section 4.2.2.

Table 3.6: Structure of the EDLSTM model in this study.

Model	Layer	Output shape	Number of parameters
EDLSTM	LSTM	(None, 95, 128)	67,584
	LSTM	(None, 128)	131,584
	RepeatVector	(None, 95, 128)	0
	LSTM	(None, 95, 64)	49,408
	TimeDistributed(Dense)	(None, 95, 64)	4,160
	TimeDistributed(Dense)	(None, 95, 1)	65

### 3.4.4 Combination of CNN and RNN

Donahue et al. (2015) combined CNN and LSTM to process a sequence of images. The visual features are first extracted using CNN and then fed into LSTM to give predictions. Nowadays, the idea of combining CNN and LSTM is getting more noticeable in time series prediction as well. The motivation behind it is that the 1-dimensional CNN layers focus on local areas and extract high-level features. Then, LSTM layers can obtain these features and capture the long-term dependency.

There are two common ways to combine CNN and LSTM for time series prediction. First, we can deal with the problem as a sequence-to-sequence problem (Fig. 3.16a). We keep the sequence length the same as we want with the help of the padding option. After several 1D CNN layers, we connect the outputs to the RNN layers, which means the different channels will be considered as different features in the RNN layers. Since the high-level features extracted by the CNN layers usually do not contain all information of the whole sequence, the bidirectional RNN is preferred for our purpose. The other idea is combining the CNN layers

with the Encoder-Decoder structure (Fig. 3.16b). First, some high-level features are extracted from several 1D CNN layers. Then, the outputs of the CNN layers will be flattened into a 1D vector, which serves as the context vector in the Encoder-Decoder structure as shown in Fig. 3.15.

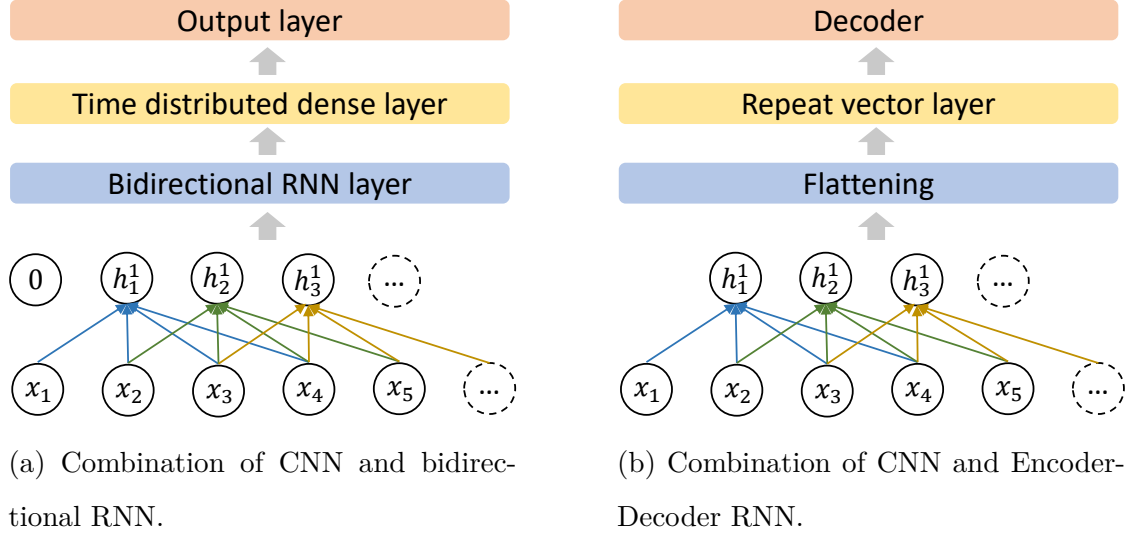


Figure 3.16: The two strategies to combine CNN and RNN.

The central difficulty for us to apply the second idea to combine CNN and RNN is that the dimension of the context vector after flattening is too high for an RNN Decoder. Since we did not apply any pooling layers, the CNN layers do not reduce the sequence length significantly, and the length of the vector after flattening will be sequence length times the number of channels, which is a huge number. Therefore, we choose to use the first idea to combine CNN and RNN. The model used in this study is called CNNLSTM since we use the LSTM neurons. The structure of the used CNNLSTM model is reported in Table 3.7.

### 3.4.5 Optimization strategy

Hyperparameter tuning is one of the most critical steps of DL algorithms. After some initial tests, we decided on some potential selections of batch size, loss function, optimizer, and learning rate. Then, a grid-search strategy has been implemented by using *Weights and Biases* (Biewald, 2020) to find the best combination of the hyperparameters. An example of the hyperparameter tuning results of the CNNLSTM model is shown in Fig. 3.17.

Based on these results, we decided to use the following combination of hyperparameters:

Table 3.7: Structure of the CNNLSTM model in this study.

Model	Output shape	Number of parameters	
CNNLSTM	Conv1D	(None, 95, 4)	52
	Conv1D	(None, 95, 8)	136
	Conv1D	(None, 95, 16)	528
	Bidirectional(LSTM)	(None, 95, 60)	11'280
	Bidirectional(LSTM)	(None, 95, 25)	8'400
	Bidirectional(LSTM)	(None, 95, 14)	2'016
	TimeDistributed(Dense)	(None, 95, 1)	15

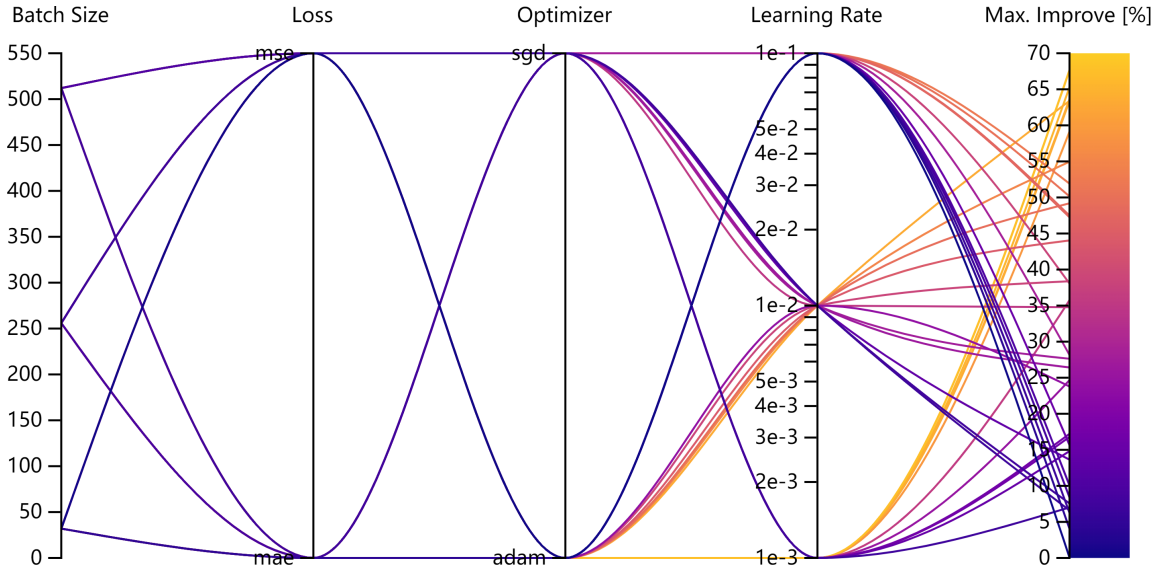


Figure 3.17: The grid-search results of hyperparameter tuning of the CNNLSTM model

- Batch size: 512
- Loss function: mean absolute error (MAE)
- Optimizer: Adam (Kingma and Ba, 2014)
- Learning rate: 0.001

The MAE loss function means:

$$\arg \min_W \sum_{i=1}^N |y_{i,\text{pred}} - y_{i,\text{true}}| \quad (3.21)$$



where  $N$  is the number of training samples. The MAE loss function is more robust against outliers than the mean squared error and therefore shows superiority in our purpose.

## 3.5 Feature Inclusion and Grouping

In this section, the inclusion of additional features (besides standardized RTN error components) and the grouping of orbit data is introduced. These experiments are carried out to analyse their influence on the results of the orbit predictions.

### 3.5.1 Feature Inclusion

In Chapter 3.2.2, the main features involved for the machine learning approaches, the standardized RTN errors, are presented. In addition to this, further features are included in order to analyse their effect on the orbit predictions. As discussed in Chapter 2.2, solar activity influences the movement of the satellites on their orbits. It thus seems reasonable to include solar parameters as additional features to potentially improve the machine learning results. The solar parameters introduced in Chapter 2.2 (Solar Flux,  $K_p$  index,  $C_{0,0}$ ) are all added individually as additional features to the input data. The matrix (3.22) shows a simplified structure of the input matrix with the RTN errors and potential additional features.

$$\begin{pmatrix} R\_error_1 & T\_error_1 & N\_error_1 & feature1_1 & \cdots \\ R\_error_2 & T\_error_2 & N\_error_2 & feature1_2 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \\ R\_error_m & T\_error_m & N\_error_m & feature1_3 & \cdots \end{pmatrix} \quad (3.22)$$

Further experiments are carried out with the  $C_{0,0}$  terms of ionosphere. Besides adding the terms as a single additional feature, the beta angles of solar position are included as a second additional feature. Moreover, the  $C_{0,0}$  terms of ionosphere are projected separately on the RTN components depending on the solar positions.

### 3.5.2 Grouping

In addition to the inclusion of further features, different groupings of orbit data are tested. Thus it is to be analysed whether different groupings of the satellite data lead to different results of the prediction errors. The first criterion after which the data is grouped is the orbital planes. GPS satellites are arranged in six orbital planes, named plane A-F, with different values of the ascending node. (Rothacher et al., 2021) Figure 3.18 shows exemplarily the constellation according to the orbital planes of the GPS satellites on the 1st of September 2021. Satellites of the same plan thus have similar orbital conditions. To group the satellite data according to their planes could have a positive effect on the orbit prediction.

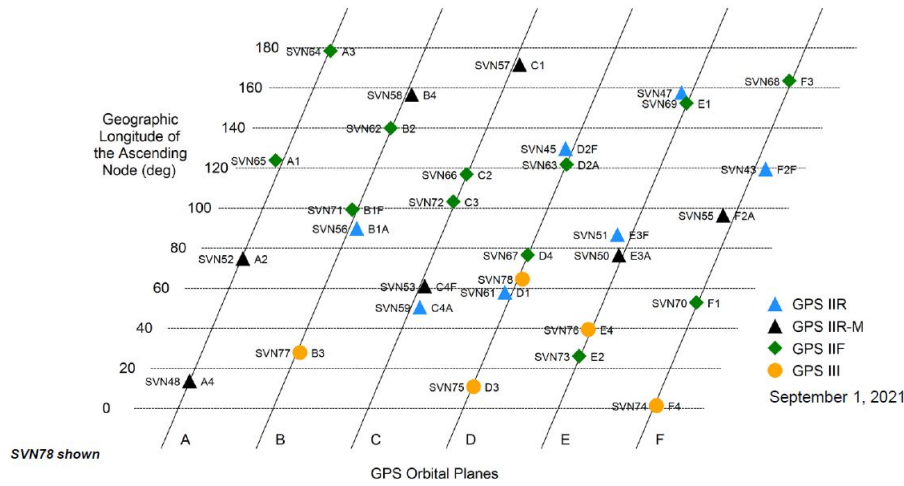


Figure 3.18: Example of GPS constellation (U.S. Coast Guard Navigation Center, 2021)

Another criterion to group the data is the block in which they were launched. GPS satellites are launched in blocks, depending on the generation they belong to (Rothacher et al., 2021). Figure 3.18 shows the allocation of the individual satellites to the respective block with different colours (green, red, blue). From block generation to block generation, the satellites have been further developed and optimised. Grouping by blocks could therefore have a positive influence on the orbit prediction since satellites in a block have a similar stage of development and thus similar characteristics.

### 3.6 Evaluation metrics

In order to evaluate the performance of our methods numerically, we introduce some evaluation metrics in this section. Since our task is time series prediction, we will evaluate our results in two aspects: the relationship between the averaged performance of all the samples and time and the averaged performance over time of each sample. The evaluation metrics are defined as following:

$$\tilde{y}_{i,t} = y_{i,t} - \hat{y}_{i,t} \quad (3.23)$$

$$\text{RMS}_t = \sqrt{\frac{\sum_{i=1}^N \tilde{y}_{i,t}^2}{N}} \quad (3.24)$$

$$\text{RMS}_i = \sqrt{\frac{\sum_{t=1}^S \tilde{y}_{i,t}^2}{S}} \quad (3.25)$$

$$\text{PR} = \frac{\text{Num} \left\{ \text{RMS}_i < \sqrt{\frac{\sum_{t=1}^S y_{i,t}^2}{S}} \right\}}{N} \cdot 100\% \quad (3.26)$$

where  $y$ ,  $\hat{y}$ , and  $\tilde{y}$  denote the true errors, predicted errors using the models, and reduced errors, respectively. The subscripts denote the time epoch  $t$  of the  $i^{\text{th}}$  sample.  $N$  and  $S$  are the number of samples and sequence length (95), respectively. Therefore,  $\text{RMS}_t$  describes the averaged performance over all samples at the individual time epoch  $t$ , whereas  $\text{RMS}_i$  describes the averaged performance over all time steps of the  $i^{\text{th}}$  sample. The positive ratio (PR) describes the proportion of samples that our approach provides positive improvement.

# Chapter 4

## Results and Discussion

In this section the results are presented and discussed. Firstly, the performance of the machine learning approaches are analysed, followed by the performances of the deep learning approaches.

### 4.1 Machine Learning

In the first subsection an overview of the ML results is given. Subsequently, the performance of the individual algorithms is discussed, followed by an analysis of the experiments of feature inclusion and grouping.

#### 4.1.1 Results Overview

In this section, the results of the five applied ML algorithms, namely linear regression, decision tree, random forest, support vector machine, and multi layer perceptron, are discussed. First, an overview of the results is given, followed by a discussion of the performance of the individual algorithms. Figure 4.1 shows the RMS w.r.t. propagation time of the improved orbit computed by the ML algorithms compared to the original ultra-rapid data. In addition to this, Table 4.1 contains the statistics of the absolute improvements.

It is visible in Figure 4.1 that MLP outperforms the other algorithms significantly. The radial component (top left) shows a clear 12h periodicity. Linear Regression provides surprisingly good results, which are comparable to those of RF. As expected, DT performs worse than RF, but still shows significant improvements. Remarkably worse results are delivered by SVM,

which contains numerous jumps along with the time interval. Similar behaviour is visible for the tangential component (top right). In this case, MLP shows even better improvements in comparison to the other algorithms, especially with increasing propagation time. This implies that the accumulated tangential orbit errors can be grasped best by MLP. The most challenging component to model is the normal component (bottom left). In contrast to the other two, the normal component doesn't feature a detectable periodicity like the radial component, nor a similar error accumulation like the tangential component. None of the algorithms applied clearly outperforms the others, however, MLP and linear regression show slightly better results than RF and DT. SVM is not capable to improve orbit errors and exhibits a large number of jumps. Considering the 3D RMS (bottom right) it can be stated, that the majority of the applied ML algorithms shows significant improvements.

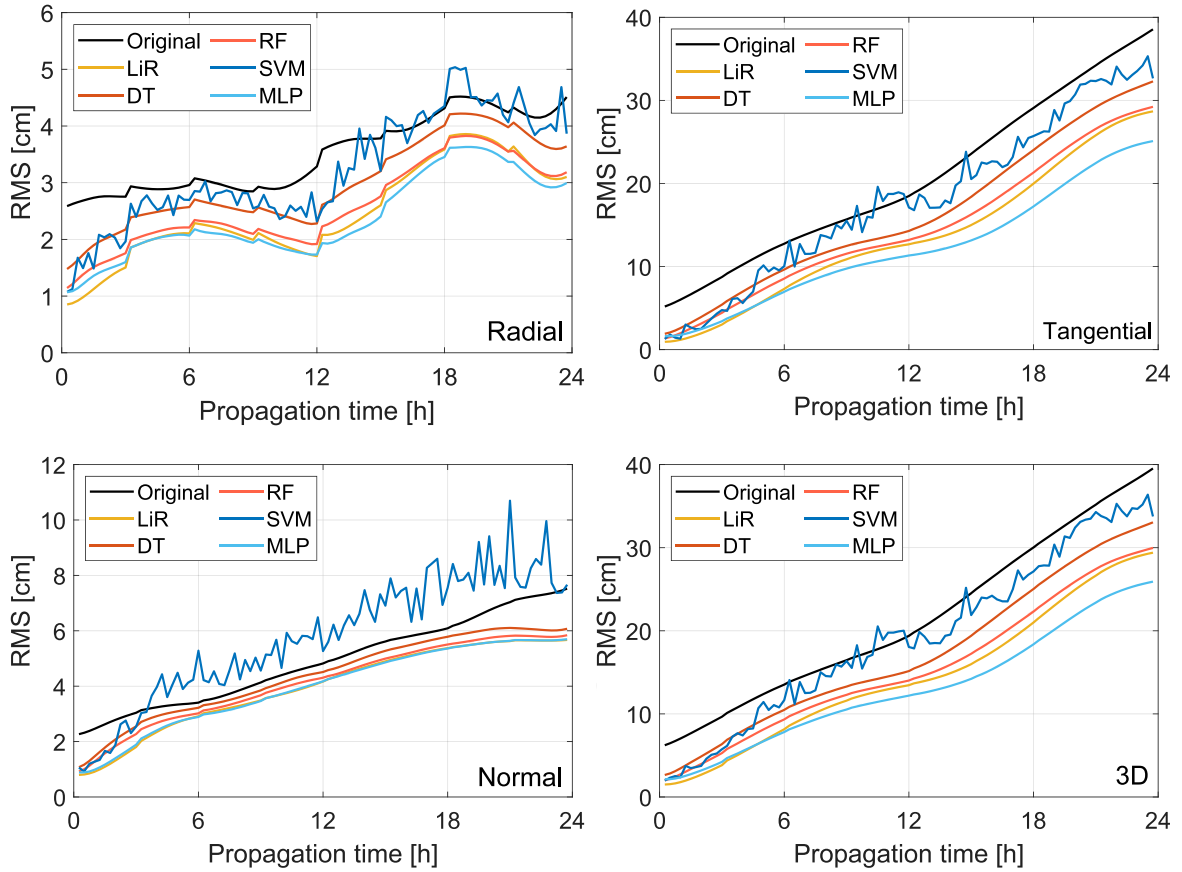


Figure 4.1: The RMS w.r.t. time of the five ML models and the original ultra-rapid products. The four sub-figures show the Radial RMS, Tangential RMS, Normal RMS, 3D RMS, respectively.

The superiority of MLP is also reflected in the statistics shown in Table 4.1, where the absolute improvements w.r.t. time are displayed in centimeters. Apart from one exception MLP consistently delivers the best results. Especially to mention is the performance of this algorithm regarding the average result of the tangential component. This demonstrates that MLP best captures the strong accumulation of the tangential orbit errors. Linear regression provides good results as well, followed by random forest and decision tree. Less promising results are delivered by SVM, which is especially visible in the normal component.

Table 4.1: The maximal, averaged, minimal absolute improvements w.r.t. time in centimeter of the five models. The best performances are highlighted.

	Radial			Tangential			Normal			3D		
	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.
LiR	1.1	0.4	0.2	13.1	4.6	2.0	<b>1.5</b>	0.5	<b>0.4</b>	13.1	5.4	2.2
DT	0.9	0.5	0.1	9.0	3.7	1.7	1.1	0.5	0.2	9.0	3.8	1.7
RF	1.1	0.5	0.2	12.7	5.0	1.9	1.4	0.7	0.2	12.8	5.0	2.0
SVM	0.7	0.3	0.1	6.7	3.2	0.8	0.0	-0.4	-0.6	6.7	3.3	1.0
MLP	<b>1.3</b>	<b>0.8</b>	<b>0.3</b>	<b>14.2</b>	<b>9.1</b>	<b>5.0</b>	1.4	<b>1.0</b>	<b>0.4</b>	<b>14.3</b>	<b>9.1</b>	<b>5.3</b>

Table 4.2 shows the positive ratios of the applied ML algorithms. The radial component can be captured best, while the tangential component is more challenging to improve. The accumulation of orbit errors in the tangential component is responsible for this. Overall, MLP shows the highest positive ratios, followed by linear regression, RF, and DT.

Table 4.2: The positive ratio of the five ML models [%].

Model	Radial	Tangential	Normal	3D
LiR	78	53	73	58
DT	73	61	67	67
RF	83	66	72	73
SVM	45	22	28	22
MLP	81	66	73	74

To analyse the results provided by ML algorithms in more depth, Figure 4.2 shows box-plots of the relative improvements of individual samples w.r.t. the original products in percentage. The absolute improvements can be found in Figure A.1. The representation of the performance emphasises the superiority of MLP once more. Interesting to see is that linear regression shows good median improvements, as already seen in Figure 4.1, the data however is distributed over a significantly wider range compared to DT, RF, and MLP. SVM again shows rather unpromising results.

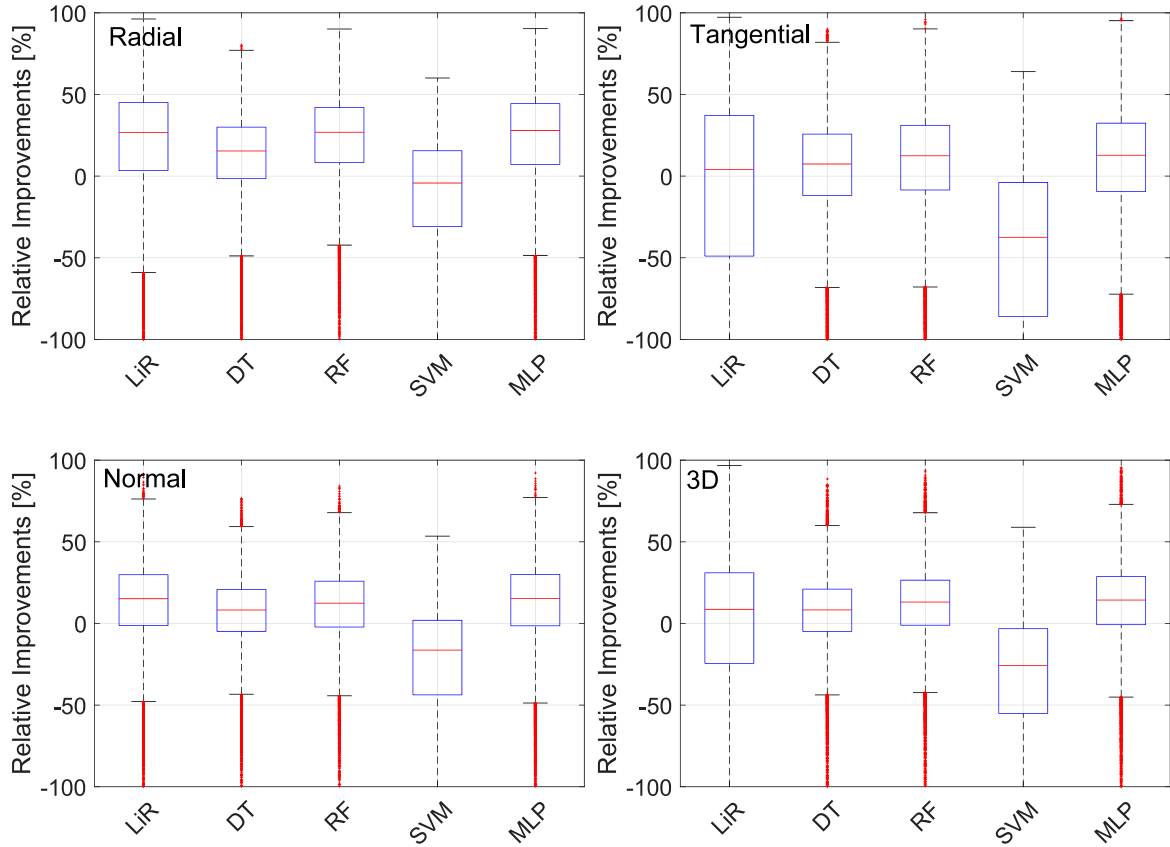


Figure 4.2: The relative improvements of individual samples using the five ML models w.r.t. the original ultra-rapid products in percentage. The four sub-figures show the Radial improvements, Tangential improvements, Normal improvements, 3D improvements, respectively.

#### 4.1.2 Discussion about Linear Regression

As seen in the previous section, linear regression shows promising results for orbit error prediction. Although this is the simplest ML algorithm applied in this study, the performance

of linear regression comes close to the one of the deep learning algorithm MLP. The positive ratio of the tangential component obtained by linear regression however shows significantly less performance than MLP, since this algorithm better captures the error accumulation of this component. The modelled improvements are distributed over a wide range for the tangential and the radial component, cf. Figure 4.2.

The results for the best performing parameters obtained by bayesian search correspond to the default settings of the algorithm provided by scikit-learn (Pedregosa et al., 2011). The used parameters are listed below:

- `fit_intercept`: *True (intercept is calculated for model)*
- `normalize`: *False (regressor not normalized before regression)*

In conclusion it can be said that linear regression shows promising results despite the simple linear approach of this algorithm.

### 4.1.3 Discussion about Decision Tree

Also, the rather simple ML algorithm DT shows significant improvements in orbit error prediction. The results do not come up to those of MLP, but overall provide useful results. For this algorithm as well, the normal component is most challenging to model. Nevertheless, DT is able to capture the main trend of this orbit error component. The distribution of the data is rather compact and demonstrates certain stability in the prediction of this algorithm, cf. Figure 4.2.

As already mentioned in Section 3.3.2, also for this model the predefined algorithm from scikit-learn (Pedregosa et al., 2011) is used. The parameter settings obtained by bayesian search are listed below:

- `max_depth`: *50 (depth of the tree)*
- `min_samples_split`: *2 (min number of samples required to split)*
- `min_samples_leaf`: *18 (min number of samples required at leaf node)*
- `max_features`: *auto (number of features considered)*

Overall, DT shows promising results and a certain stability in the prediction throughout all components.



#### 4.1.4 Discussion about Random Forest

As expected, RF performs better than DT. The results are nevertheless comparable and show similar behaviours. RF throughout provides promising improvements, even the trend of the tangential component is clearly captured. Of the ML algorithms applied in this study, only MLP outperforms RF regarding the positive ratio, cf. Table 4.2. Comparable to DT, the distribution of the data is rather compact, cf. Figure 4.2.

The best performing parameters of RF, obtained by bayesian search, are listed below:

- `n_estimators`: *125 (number of trees in the forest)*
- `max_depth`: *23 (depth of the tree)*
- `min_samples_split`: *2 (min number of samples required to split)*
- `min_samples_leaf`: *2 (min number of samples required at leaf node)*
- `max_features`: *auto (number of features considered)*

In summary it can be said, that RF is able to provide significant orbit error improvements. As well as DT, the algorithm demonstrates stability throughout the prediction of all components.

#### 4.1.5 Discussion about Support Vector Machine

SVM provides the least fitting results. The RMS w.r.t. time shows numerous jumps and outliers along the time span, cf. Figure 4.1. Throughout all components and ratios, SVM shows the worst performance compared to the other ML algorithms applied on the orbit data. Also for SVM the normal component is the most challenging one to model, where almost no improvement is achieved in this case, cf. Table 4.1.

The chosen hyperparameters used for this model, which are obtained by bayesian search, are listed below:

- `epsilon`: *0.12 (range of decision boundary)*
- `kernel`: *rbf (kernel function)*
- `max_iter`: *180 (number of iterations)*

In the context of this study, SVM could not show promising results. Overall, SVM can slightly improve the orbit prediction errors for the radial and the tangential component, however, numerous jumps and outliers appear in the time series, cf. Figure 4.1. The performance of SVM is rather surprising compared to the performances of the other algorithms applied, since SVM is known as a robust and accurate ML approach (Suthaharan, 2016). In a study of Peng and Bai (2018a) it has been shown, that SVM is able to improve the orbit error accuracy by capturing the underlying pattern with good performance. Unfortunately, this is not the case in this study. There is no obvious reason to explain the behaviour of the performance of SVM in this study. However, a crucial parameter, which has to be set for SVM, is the kernel function. An unappropriated kernel function can lead to bad performance of the algorithm (Han et al., 2012). Although the parameter search was carried out with bayesian search and therefore the best performing values are traced, unappropriated parameter settings may lead to the rather bad performance of SVM.

#### 4.1.6 Discussion about Multi Layer Perceptron

As already indicated in Section 4.1.1, MLP shows promising results throughout the analysis and outperforms the other algorithms to a fair extent, cf. Table 4.1. It is to be expected, that MLP shows superiority in this framework since this algorithm belongs to the class of deep neural networks and not the basic ML algorithms. Particularly noteworthy is the absolute average improvement of the tangential component w.r.t. time, cf. Table 4.1. This value exceeds the ones achieved from the other ML algorithms by almost double.

The parameter settings used in this study, obtained by bayesian search, are listed below:

- `hidden_layer_sizes`: *(23,16,9) (number of hidden layers and sizes)*
- `activation`: *relu (activation function)*
- `solver`: *adam (solver for weight optimization)*
- `max_iter`: *68 (number of iterations)*

Thus it can be said, that MLP shows highly promising results. The performance of MLP is stable throughout the process and shows high positive ratios, cf. 4.2.

### 4.1.7 Discussion about Feature Inclusion and Grouping

As introduced in Section 3.5.1, parameters describing solar activity are incorporated in the process of ML to analyse their influence on the orbit error prediction. Three different parameters, namely 10.7cm flux,  $K_p$  index, and  $C_{0,0}$  term of ionosphere, are separately incorporated as additional features. Here, the results of the feature inclusion using the best performing ML algorithm MLP (cf. 4.1.1) are shown. The different experiments of inclusion of solar parameters are listed below:

- F1: Include *10.7 cm Solar Flux* as additional feature
- F2: Include  $K_p$  *index* as additional feature
- F3: Include  $C_{0,0}$  *term of ionosphere* as additional feature

The statistics of the three experiments in comparison to the basic MLP approach are shown in Table 4.3. None of the included features leads to significant improvements in the orbit error prediction. A possible cause of this is that all three solar data sets are expanded with linear interpolation due to temporal resolution of the orbit data, cf. 2.2. This process may influence the data sets in a way that the ML algorithms cannot benefit from the feature inclusion.

Table 4.3: The maximal, averaged, minimal absolute improvements w.r.t. time in centimeter of MLP and MLP with inclusion of solar parameters. The best performances are highlighted.

	Radial			Tangential			Normal			3D		
	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.
MLP	<b>1.3</b>	<b>0.8</b>	<b>0.3</b>	<b>14.2</b>	9.1	5.0	<b>1.4</b>	<b>1.0</b>	<b>0.4</b>	<b>14.3</b>	9.1	<b>5.3</b>
F1	1.2	<b>0.8</b>	0.2	14.0	<b>9.3</b>	4.9	1.3	<b>1.0</b>	0.3	14.1	<b>9.3</b>	5.0
F2	<b>1.3</b>	0.7	<b>0.3</b>	14.1	8.7	<b>5.1</b>	1.1	0.9	<b>0.4</b>	14.2	8.9	<b>5.3</b>
F3	<b>1.3</b>	<b>0.8</b>	0.2	13.5	9.1	5.0	<b>1.4</b>	<b>1.0</b>	<b>0.4</b>	13.6	9.1	5.2

In further analysis, we show that the inclusion of  $C_{0,0}$  term of ionosphere is able to contribute to the orbit error prediction, cf. 4.2.3.

Next to feature inclusion, we tried different data groupings to analyse their influence on the orbit error prediction, cf. 3.5.2. The grouping of data according to the six orbital planes and the block generation does not show any improvements in orbit error prediction. These two criteria are thus not conducive within the context of ML approach.

## 4.2 Deep Learning

### 4.2.1 Results Overview

In this section, we present and discuss the results of the four basic DL algorithms, namely DNN, CNN, GRU, and LSTM. The results using advanced structures of RNN are reported and discussed in Section 4.2.2.

Fig. 4.3 show two examples of our data and the corresponding predictions using LSTM. These results prove that our proposed method can provide good predictions on both kind of orbits with (Fig. 4.3 b)) or without (Fig. 4.3 a)) strong error accumulation.

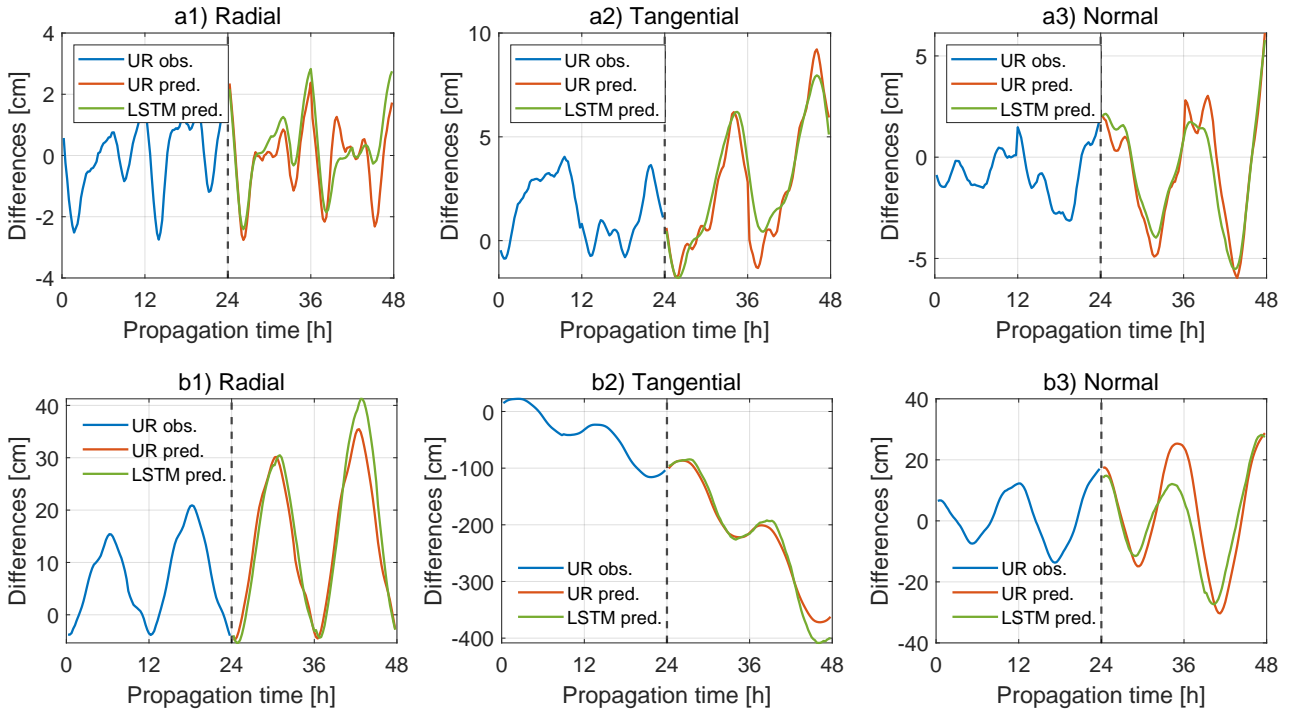


Figure 4.3: Two selected samples with (down) and without strong error accumulation (up) and the corresponding prediction results using LSTM.

Fig. 4.4 shows the behaviour of  $\text{RMS}_t$  w.r.t. propagation time of the improved orbit using the four DL models compared to the original ultra-rapid products. The statistics of the absolute improvements are shown in Table 4.4. All the four models show significant improvements. Comparing the three components we can find that the periodicity of 12h is clear in the radial components (Fig. 4.4 a)). The performances of GRU and LSTM are very similar, and both of them show clear superior to DNN and CNN. In Fig. 4.4 b) we can see that LSTM clearly outperforms GRU, especially when the propagation time is relatively long, meaning that LSTM can capture the strong accumulation in the tangential orbit errors better. This demonstrates the advantage of LSTM in solving the long-term dependencies. However, GRU and LSTM do not outperform others clearly in the normal errors (Fig. 4.4 c)). This may be because the normal components are the most difficult for DL models of the three, as they are neither as well-defined periodic as the radial components nor as significantly error-accumulating as the tangential components. Therefore, all the DL models can hardly capture a clear relationship between the input features and targets.

Besides, the good performance of the sequential modelling is also proved by the statistics shown in Table 4.4. All the best results are from GRU and LSTM. Among them, LSTM outperforms GRU, especially in the tangential components, which demonstrates the necessity of the complicated neuron of LSTM for our study. The LSTM neural network achieves the performance with maximal absolute improvement of  $\text{RMS}_t$  over 16.8 cm and averaged relative improvement of  $\text{RMS}_t$  over 45%. Besides, the maximal relative improvement of  $\text{RMS}_t$  is over 70% at the first epochs of the propagation. Because of the excellent performance of LSTM, later in Section 4.2.2, all the advanced RNN structures use LSTM neurons.

We also analyzed the performance of the DL models on individual samples. Since we are using DL models to predict the differences between the ultra-rapid products and the final products, sometimes the DL model could hurt the accuracy of the ultra-rapid products when they give terrible predictions. Therefore, we report the relative improvements (Fig. 4.5) and absolute improvements (Fig. A.3) of  $\text{RMS}_i$  to analyze the performance of the models on each individual samples. The positive ratios are represented in Table 4.5. The radial components are the easiest for the DL models to improve due to the periodicity, whereas the tangential components are the most difficult due to the error accumulation. Besides, all

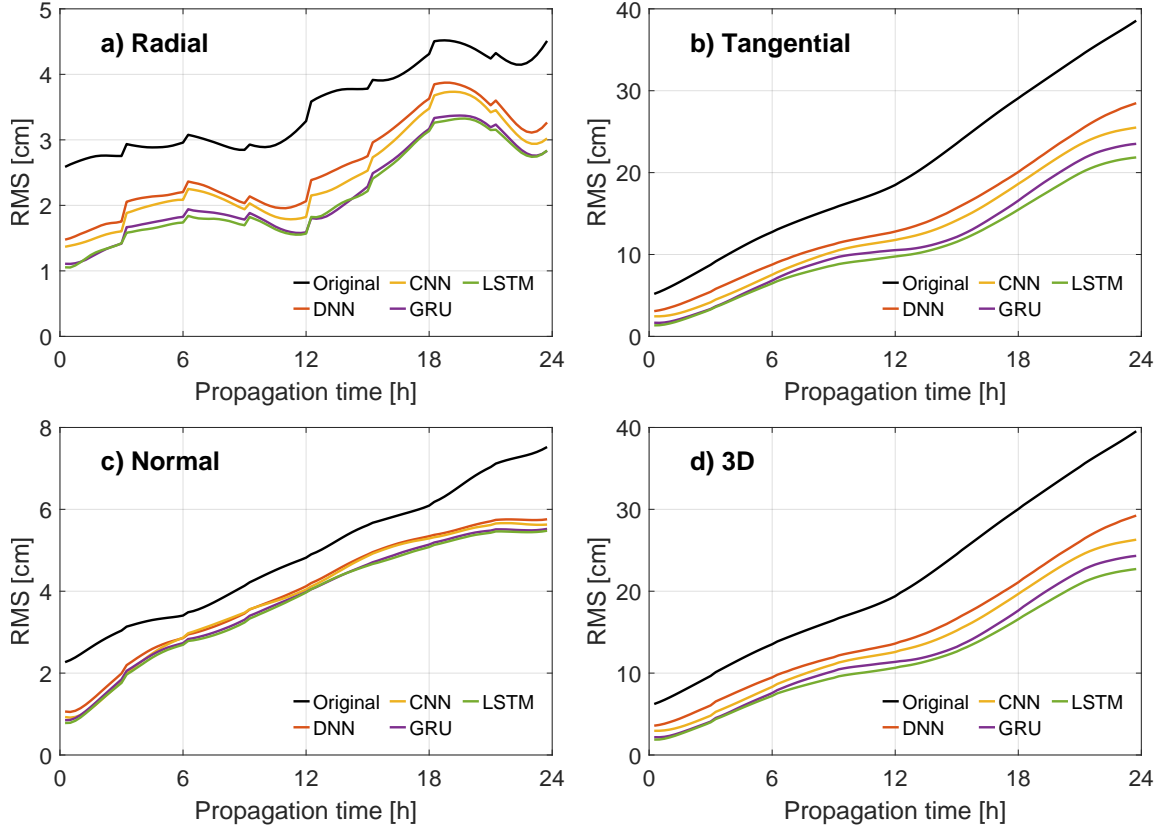


Figure 4.4: The  $RMS_t$  of the four basic DL models and the original ultra-rapid products. The four sub-figures show the a) Radial  $RMS_t$ , b) Tangential  $RMS_t$ , c) Normal  $RMS_t$ , d) 3D  $RMS_t$ , respectively.

Table 4.4: The maximal, averaged, minimal absolute improvements of  $RMS_t$  in centimeter of the four models. The best performances are highlighted.

	Radial			Tangential			Normal			3D		
	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.
DNN	1.2	0.9	0.6	10.1	6.2	2.1	1.8	0.9	0.5	10.3	6.3	2.6
CNN	1.5	1.0	0.7	13.0	7.6	2.7	1.9	1.0	0.5	13.2	7.6	3.3
GRU	<b>1.9</b>	<b>1.3</b>	1.0	15.0	8.9	3.5	<b>2.0</b>	<b>1.1</b>	0.6	15.2	9.0	4.0
LSTM	1.8	<b>1.3</b>	<b>1.1</b>	<b>16.7</b>	<b>9.7</b>	<b>3.8</b>	<b>2.0</b>	<b>1.1</b>	<b>0.7</b>	<b>16.8</b>	<b>9.7</b>	<b>4.3</b>

the four models have similar performance in the aspect of positive ratio with slightly greater median improvements of GRU and LSTM as shown in Fig. 4.5.

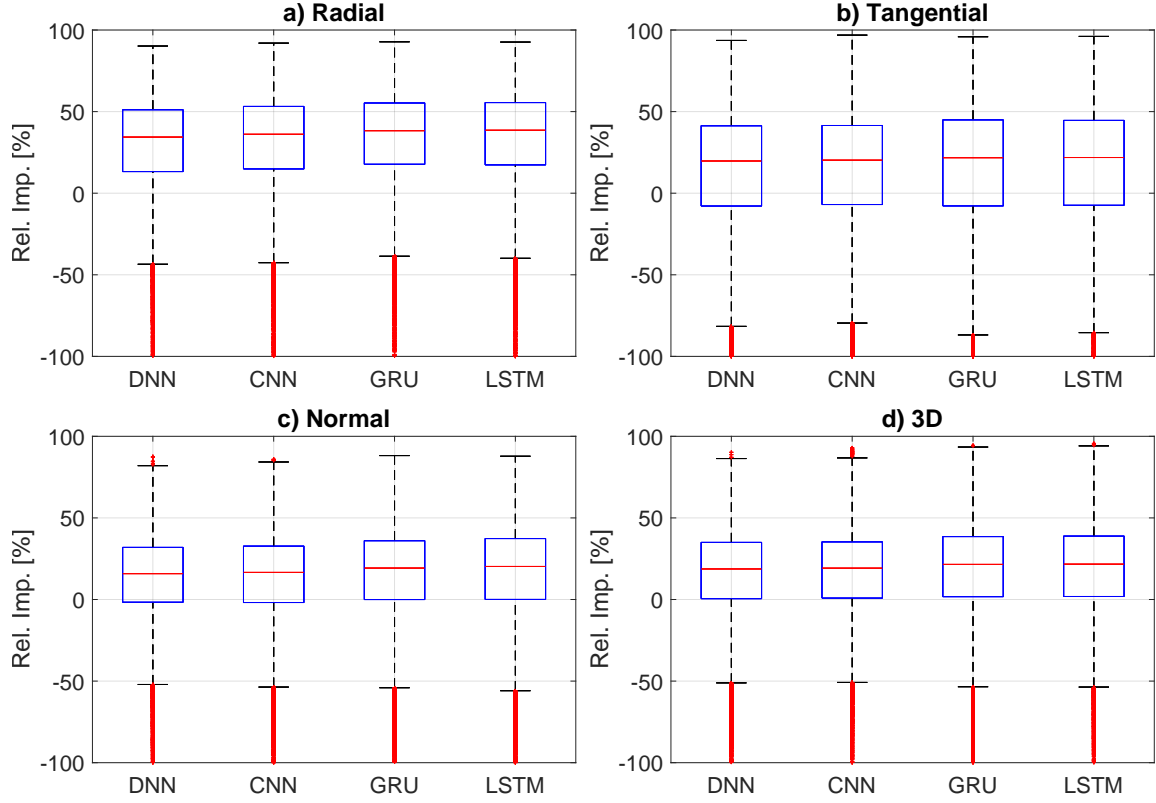


Figure 4.5: The relative improvements of  $RMS_i$  using the four basic DL models w.r.t. the original ultra-rapid products in percentage. The four sub-figures show the a) Radial improvements, b) Tangential improvements, c) Normal improvements, d) 3D improvements, respectively.

Table 4.5: The positive ratio of the four basic DL models [%].

Model	Radial	Tangential	Normal	3D
DNN	85	69	73	76
CNN	86	70	73	76
GRU	88	70	75	77
LSTM	88	70	75	77

#### 4.2.2 Discussion about RNN

As we mentioned in Section 3.4.3, we know that the original sequence-to-sequence modelling is not feasible for our purpose since we should summarize the whole input sequence. Therefore, we report the results of the more advanced RNN structures in this section. Since the results in Section 4.2.1 prove that the LSTM networks outperform GRU networks, all the structures are

combined with the LSTM neurons, namely biLSTM, EDLSTM, and CNNLSTM. Besides, the results of the original sequence-to-sequence modelling LSTM are also reported to demonstrate our expectations.

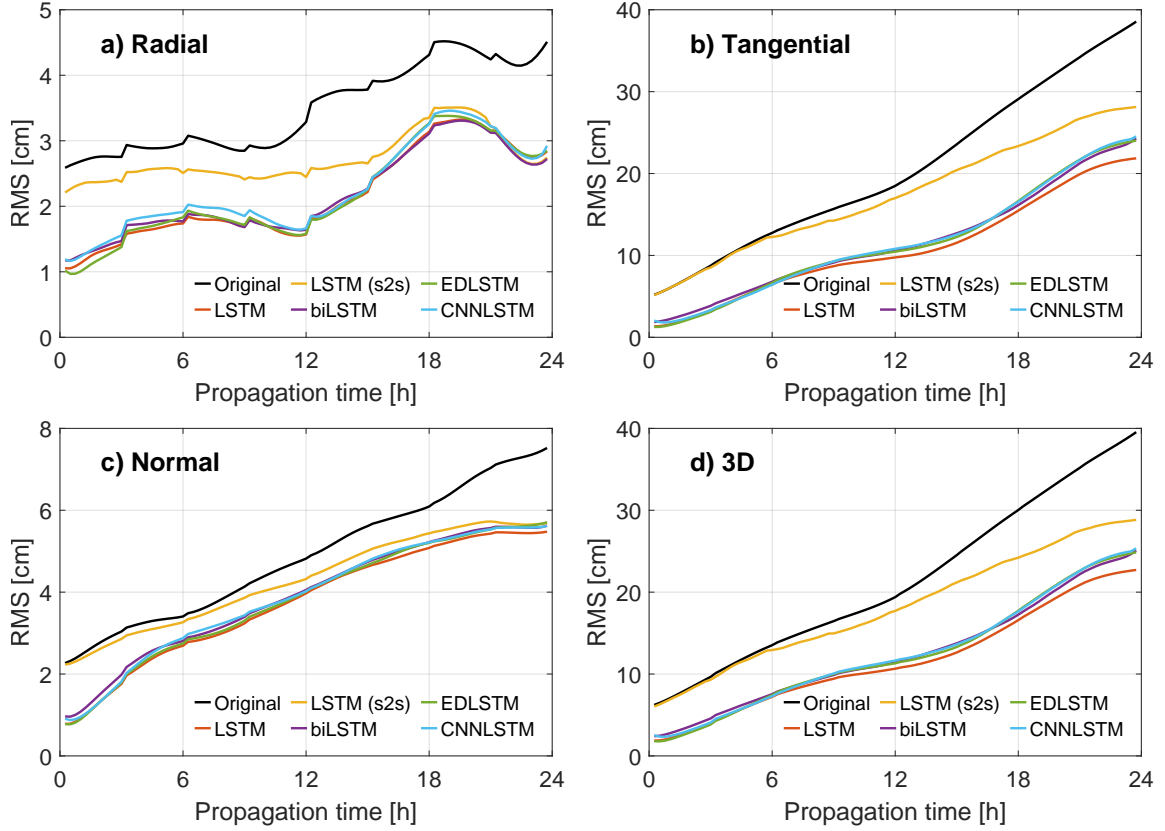


Figure 4.6: The  $RMS_t$  of the five different LSTM structures and the original ultra-rapid products. The four sub-figures show the a) Radial  $RMS_t$ , b) Tangential  $RMS_t$ , c) Normal  $RMS_t$ , d) 3D  $RMS_t$ , respectively.

Fig. 4.6 shows the behaviour of the  $RMS_t$  of the improved orbit using the five LSTM models compared to the original ultra-rapid products. The statistics of the absolute improvements of  $RMS_t$  are shown in Table 4.6. We can notice that the LSTM (s2s) performs much worse than all the other models, especially at the first epochs. As we described in Section 3.4.3 and showed in Fig. 3.11a, the outputs on the first few output epochs only contain the information on the first few input epochs, which is unreasonable for our purpose. On the other side, the outputs on the latter epochs contain more information of the input sequence. Therefore, the performance of LSTM (s2s) on the latter epochs is much better.



Table 4.6: The maximal, averaged, minimal absolute improvements of  $\text{RMS}_t$  in centimeter of the five different LSTM structures. The best performances are highlighted.

	Radial			Tangential			Normal			3D		
	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.
LSTM	1.8	<b>1.3</b>	<b>1.1</b>	<b>16.7</b>	<b>9.7</b>	3.8	<b>2.0</b>	<b>1.1</b>	<b>0.7</b>	<b>16.8</b>	<b>9.7</b>	4.3
LSTM (s2s)	1.8	0.8	0.3	10.4	3.1	0	1.8	0.6	0	10.7	3.3	0.2
biLSTM	1.8	<b>1.3</b>	<b>1.1</b>	14.3	8.9	3.3	1.9	1.0	0.6	14.5	8.9	3.8
EDLSTM	<b>1.9</b>	<b>1.3</b>	1.0	14.5	9.0	<b>3.9</b>	1.8	<b>1.1</b>	0.6	14.7	9.0	<b>4.4</b>
CNNLSTM	1.8	1.2	0.9	14.0	8.9	3.2	1.9	1.0	0.5	14.2	8.9	3.7

Besides, we can also notice that the advanced variants of LSTM do not provide significant improvements. The possible reason is the number of trainable parameters. As mentioned in Section 2.1, we have 144'075 training samples in total. However, all the advanced structures (biLSTM, EDLSTM, CNNLSTM) have more trainable parameters. Although the DL models can usually keep good generality with more trainable parameters than training samples, they may perform worse for certain complicated tasks where the relationship between the input features and outputs is not very strong. In these cases, the models with more trainable parameters will suffer overfitting problems. In our study, all the DL methods suffer the overfitting issue, which can be observed when comparing the training and validation loss. Here we give the examples of the validation (training) loss of the tangential components:

- LSTM: 0.2698 (0.2242)
- LSTM (s2s): 0.3761 (0.3490)
- biLSTM: 0.2766 (0.2163)
- EDLSTM: 0.2717 (0.2020)
- CNNLSTM: 0.2627 (0.2482)

We can see that biLSTM and EDLSTM suffer the overfitting issue more strongly than LSTM because they have much smaller training loss but higher validation loss. This indicates that these two models are more powerful than the LSTM model to capture the relationship between

the input features and outputs. However, the insufficient number of data is the reason that limits their performance. We applied different strategies to reduce the overfitting issue, such as including the *Dropout* layers or applying the regularization. However, none of those strategies solve the overfitting issue efficiently. CNNLSTM has an even better validation loss than LSTM but slightly worse results. This is a typical problem in the optimization process of DL models in that the optimizing target is different from our final target. In order to solve this problem, some specific costuming loss function is needed.

Fig. 4.7 shows the relative improvements of  $\text{RMS}_i$  using the five LSTM variants with the positive ratio shown in Table 4.7. They demonstrate again that the sequence-to-sequence structure is infeasible for our purpose and all the other methods provide similar results. However, we can also observe that the CNNLSTM model provides a slightly higher positive ratio and the under boundary of the box plot is better than -50 %, which indicates that extracting high-level features can make our predictions more stable. Once we can include more training data, this structure is promising.

Table 4.7: The positive ratio using the five different LSTM variants [%].

Model	Radial	Tangential	Normal	3D
LSTM	88	70	75	77
LSTM (s2s)	81	63	68	70
biLSTM	89	70	74	77
EDLSTM	88	70	75	77
CNNLSTM	88	72	75	78

### 4.2.3 Influences of additional features

As mentioned in Chapter 1, the major errors of GNSS orbits are expected to be caused by the deficiency of solar radiation pressure models. Therefore, in the next step, we investigate the influence of solar activities. Since we do not have direct measurements of the solar pressure, we use different related parameters as additional features to implicitly model the solar impact, as described in Section 2.2.

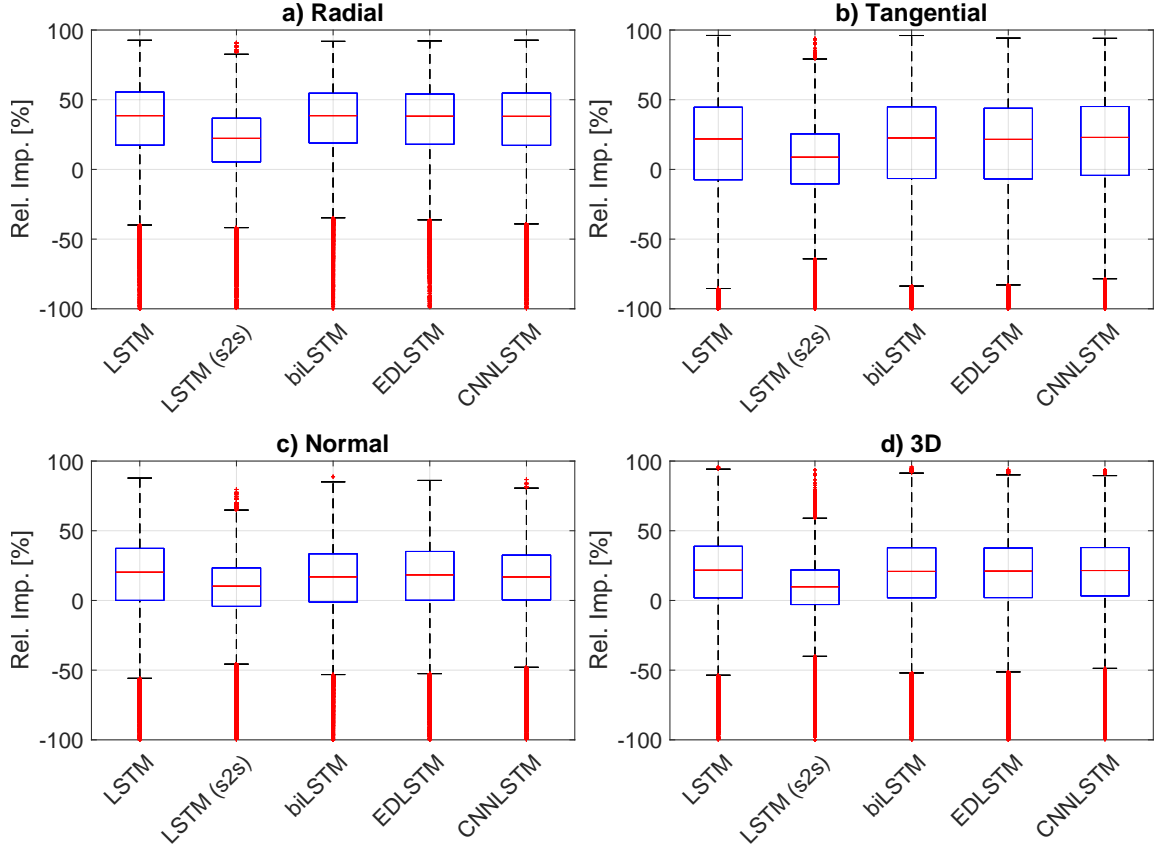


Figure 4.7: The relative improvements of  $\text{RMS}_i$  using the five different LSTM structures w.r.t. the original ultra-rapid products in percentage. The four sub-figures show the a) Radial improvements, b) Tangential improvements, c) Normal improvements, d) 3D improvements, respectively.

We have done many tests based on the different parameters. Our analysis shows that the inclusion of the 10.7 cm flux data and  $K_p$  index cannot improve the results due to their low quality and low temporal resolution. The linear interpolation is necessary for us to have the same temporal resolution as the orbit data, but it may also introduce some artificial signal, which can hurt the performance of the DL models. The  $C_{0,0}$  terms of global ionosphere maps, however, have some contribution due to the higher quality and temporal resolution. Although the linear interpolation is still necessary, the original 1-hourly resolution is much better than 3-hourly ( $K_p$  index) and daily (10.7 cm flux). Here we report three selected experiments based on the LSTM model since it performs the best among all the DL models. The settings of the three experiments are:

- S1: Include  $C_{0,0}$  (within the prediction interval) as additional features.

- S2: Compute the projection of  $C_{0,0}$  (within the prediction interval) on the corresponding component using the solar position data. Include the projected values as additional features.
- S3: Compute  $\Delta C_{0,0}$  which is the difference between  $C_{0,0}$  within the prediction interval and  $C_{0,0}$  within the observation interval. Project  $\Delta C_{0,0}$  on the corresponding components using solar position data. Include the projected values as additional features.

The behaviours of the  $\text{RMS}_t$  of the improved orbit in different experiments are shown in Fig. 4.8 with the statistics shown in Table 4.8. We can notice that in experiment S2, the tangential errors are significantly improved (ca. 10 %), whereas the other two experiments do not show any improvements. Comparing S2 and the original experiment, we can find that including the  $C_{0,0}$  terms and projecting them on the according component outperforms the original settings in all the aspects except the minimal improvements of the tangential component and 3D RMS. Besides, comparing the experiments S1 and S2, we can prove the impotence of the solar position. The idea of experiment S3 is that solar activity changes are more important than themselves. However, the results are against this idea.

Table 4.8: The maximal, averaged, minimal absolute improvements of  $\text{RMS}_t$  in centimeter of the three experiments and the original experiment using LSTM. The best performances are highlighted.

	Radial			Tangential			Normal			3D		
	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.
LSTM	1.8	1.3	<b>1.1</b>	16.7	9.7	3.8	<b>2.0</b>	<b>1.1</b>	<b>0.7</b>	16.8	9.7	4.3
S1	1.8	1.3	1.0	16.0	9.2	3.8	<b>2.0</b>	<b>1.1</b>	0.6	16.2	9.3	4.3
S2	<b>2.0</b>	1.3	<b>1.1</b>	<b>18.4</b>	<b>10.5</b>	3.6	<b>2.0</b>	<b>1.1</b>	<b>0.7</b>	<b>18.5</b>	<b>10.5</b>	4.2
S3	1.8	<b>1.4</b>	1.0	15.1	9.3	<b>3.9</b>	<b>2.0</b>	<b>1.1</b>	0.6	15.3	9.3	<b>4.4</b>

We also analyze the performance on individual samples in these experiments, as shown in Fig. 4.9 with the positive ratio shown in Table 4.9. We cannot see any significant improvements in the positive ratio (only 0.5 %) or from the figure. Further analysis shows that the major reason for the superiority of S2 is that it can reduce the number of samples on which the LSTM

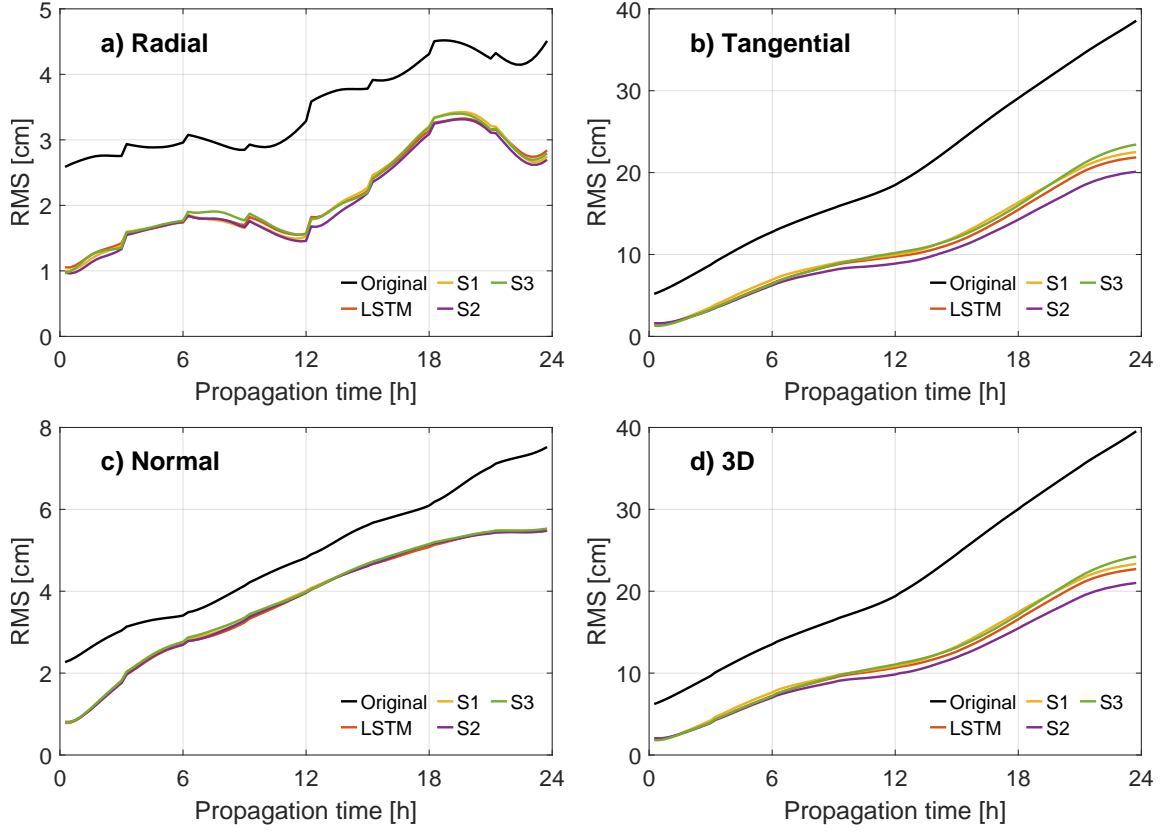


Figure 4.8: The  $RMS_t$  using LSTM with different features and the original ultra-rapid products. The four sub-figures show the a) Radial  $RMS_t$ , b) Tangential  $RMS_t$ , c) Normal  $RMS_t$ , d) 3D  $RMS_t$ , respectively.

performs terribly. If we define strongly badly predictions (SBP) whose absolute improvements under -10 cm, the number of SBP on the tangential component in the original experiment is 176, whereas in S2 is 126 (relative improvements of 28%). This indicates that including solar activity and position data can improve the performance of DL models by relieving the strongly negative impacts.

Table 4.9: The positive ratio of the four experiments [%].

Model	Radial	Tangential	Normal	3D
LSTM	88	70	75	77
S1	88	70	75	77
S2	87	70	75	77
S3	87	70	74	77

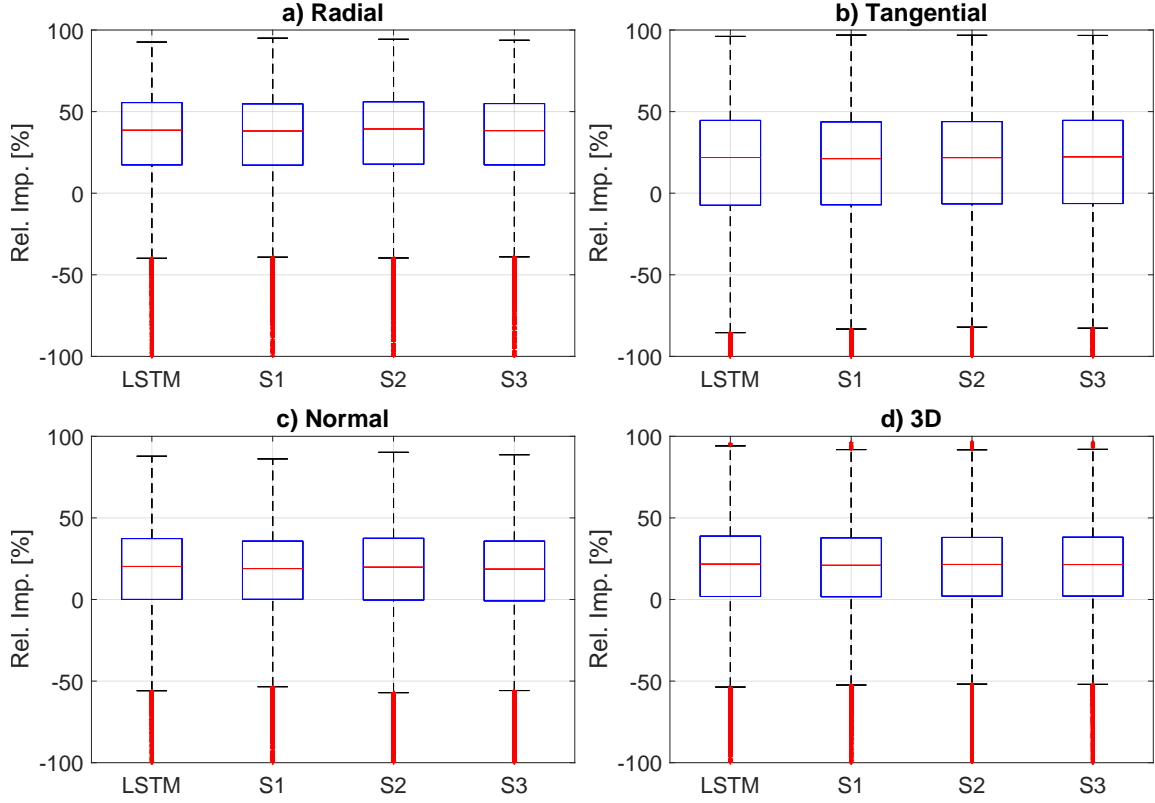


Figure 4.9: The relative improvements of  $RMS_i$  in the original experiment and the three experiments about solar activity. The four sub-figures show the a) Radial improvements, b) Tangential improvements, c) Normal improvements, d) 3D improvements, respectively.

#### 4.2.4 Other GNSS constellations

Once we demonstrate the noticeable performances of the ML/DL models on the GPS constellations, we want to apply the same strategies to the other GNSS constellations so that we can have a full set of improved orbit predictions of GNSS satellites. However, we did not find enough data for the Galileo and Beidou constellations. The GFZ IGS Analysis Center provides Galileo orbit starting from 2021 (Männel et al., 2020*a,b*). The other IGS analysis centers, such as CODE, ESA, and Wuhan University, only publicly provide GPS and GLONASS orbits. Therefore, we can only test our proposed method on the GLONASS constellation.

In this experiment, we keep the time interval of the GLONASS data the same as the other experiments about the GPS constellation. However, the number of valid data is smaller because the GLONASS constellation only has 24 satellites, and the data quality is worse than the GPS constellation. The number of training, validation, and test samples are 88'897, 19'049,

and 19'050, respectively. We applied the LSTM model since it performs the best regarding the previous analysis and keeps the same structure (Table 3.3) to avoid any additional influence.

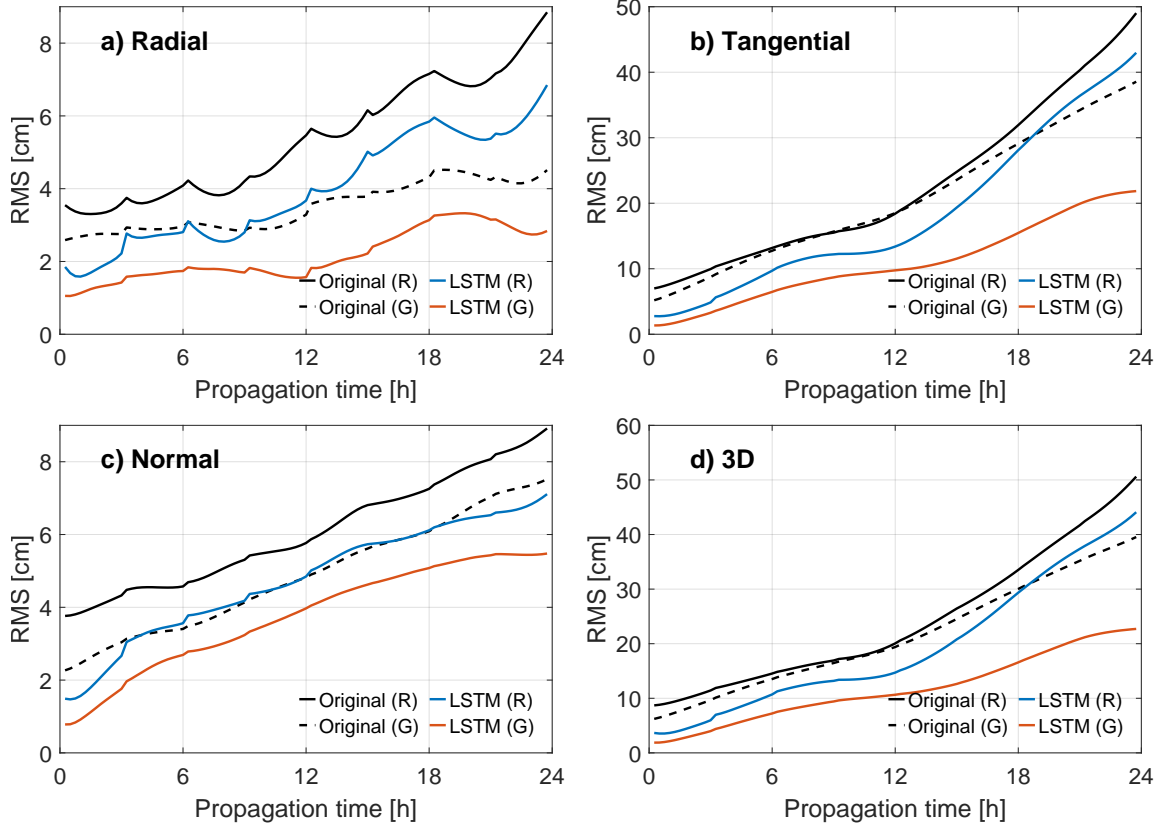


Figure 4.10: The  $RMS_t$  of the original orbits and improved orbits using LSTM of the GPS constellation (denoted by G) and GLONASS constellation (denoted by R). The four sub-figures show the a) Radial RMS, b) Tangential RMS, c) Normal RMS, d) 3D RMS, respectively.

Table 4.10: The maximal, averaged, minimal absolute improvements w.r.t. time in centimeter of GLONASS and GPS using LSTM. The best performances are highlighted.

	Radial			Tangential			Normal			3D		
	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.
GLONASS	<b>2.1</b>	<b>1.4</b>	0.9	6.0	4.4	3.1	2.3	1.3	0.8	6.5	4.8	3.5
GPS	1.8	1.3	<b>1.1</b>	<b>16.7</b>	<b>9.7</b>	<b>3.8</b>	<b>2.0</b>	<b>1.1</b>	<b>0.7</b>	<b>16.8</b>	<b>9.7</b>	<b>4.3</b>

The behaviour of  $RMS_t$  is shown in Fig. 4.11 with the statistics shown in Table 4.10. We can find that the proposed LSTM model can also improve the GLONASS orbits significantly.

However, the improvements are minor compared to the GPS constellations, especially on the tangential component and, therefore, the 3D RMS. However, we can still conclude that our proposed method can improve the quality of GLONASS orbits to the quality of GPS orbits within the first 18 hours, which is a good achievement.

Moreover, when we look at the performance on individual components, we can realize that the LSTM performs better on the GLONASS constellation with significant better median improvements of  $\text{RMS}_i$  and positive ratio (Fig. 4.11, Table 4.11), which indicates that the LSTM model provides more stable improvements on the GLONASS constellation. Further analysis shows, however, that the best performance of LSTM on individual GLONASS samples is much limited. The model provides improvement over 1 m only on 7 GLONASS samples (0.04 %), whereas on 82 GPS samples (0.3 %).

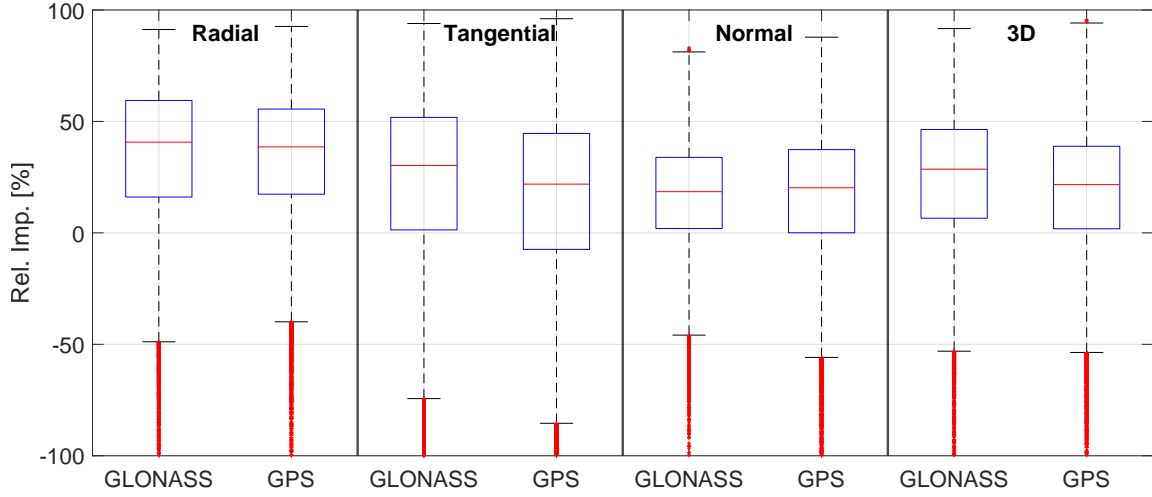


Figure 4.11: The relative improvements of  $\text{RMS}_i$  of the GLONASS and GPS constellations using LSTM. The four sub-figures show the a) Radial improvements, b) Tangential improvements, c) Normal improvements, d) 3D improvements, respectively.

Table 4.11: The positive ratio of the improved GLONASS and GPS orbits using LSTM [%].

Model	Radial	Tangential	Normal	3D
GLONASS	85	76	77	80
GPS	88	70	75	77



# Chapter 5

## Conclusion and Outlook

### 5.1 Conclusion

In this study, we applied different ML and DL algorithms to improve the accuracy of the 24-hour GNSS orbits predictions in the ultra-rapid products provided by IGS. As ground truth we consider the IGS final products, and the differences between the ultra-rapid and final products are computed. The differences within the 24-hour observations interval of the ultra-rapid products are used as input features, whereas the differences within the 24-hour prediction interval of the same ultra-rapid products serve as targets for the ML and DL models. Moreover, we also include different related parameters as additional features to implicitly model the solar impact so that we can release the deficiency of solar radiation pressure models to a certain degree. We applied our strategies on both GPS and GLONASS constellations and obtained significant improvements on both.

Our study demonstrates the promising potential of combining the physics-based orbit propagator and the data-driven ML and DL algorithms to improve the precise orbit prediction. Our analysis shows that all the selected ML and DL models, except SVM, can stably provide significant improvements. Comparing the performance of different methods, we can conclude that the neural networks outperform the other traditional ML algorithms. Besides, the RNN models dominate the study because of the time-series characteristic of the satellite orbits. The LSTM neural network achieves the best performance with a maximal absolute improvement of over 16.8 cm and averaged relative improvements over 45% in the aspect of  $RMS_t$ . Additionally, it can also provide stable improvements on more than 76% among over

30'000 test samples, which proves the generalization of the proposed method. Furthermore, our analysis shows that including the combination of  $C_{0,0}$  terms of global ionosphere maps and the solar position data as additional features can further improve the results. In the end, the improved GPS orbits keep centimeter-level accuracy over 14 hours which is 3.5 times longer than the original orbits (4 hours). The improvements are smaller on the GLONASS constellation due to the different design of orbits and also the lower quality of the final products. However, we still achieve an averaged relative improvement over 20 % in the aspect of averaged 3D RMS over time. Besides, the improvements of GLONASS are more stable with a positive ratio of higher than 80 %. In the end, the centimeter-level orbit predictions can be kept close to 6 hours for the GLONASS constellations, which means a relative improvement of 100 % compared to the original ultra-rapid predictions (3 hours).

## 5.2 Outlook

Since we have demonstrated that the GNSS satellites orbits can be improved by using ML and DL algorithms, we can apply it to other satellites. For example, the proposed methods are also promising for LEO satellites because the errors of the physics-based propagator on LEO orbits are much higher due to more significant non-gravitational perturbing forces. However, the proposed models have to be changed to consider the influence of air drags. Finding the high-quality parameters related to the non-gravitational forces like air drag or solar radiation pressures is a challenge.

Besides, we should also investigate more about GNSS constellations. First, we should find the proper data of Galileo and Beidou constellations. Then, we can also expand the time interval and obtain more training samples to enable the application of more complicated models. Once we can have the improved orbits of the full set of GNSS constellations, we can apply these improved orbits to estimate geodetic parameters. The goal will be to ensure that the improved orbits can also benefit geodetic applications. In order to evaluate the performance of our methods numerically, we introduce some evaluation metrics in this section. Since our task is time series prediction, we will evaluate our results in two aspects: the relationship between the averaged performance of all the samples and time and the averaged performance over time of each sample.

# Appendix A

## Supplementary Materials

### A.1 Absolute improvements of individual samples

#### A.1.1 Machine learning

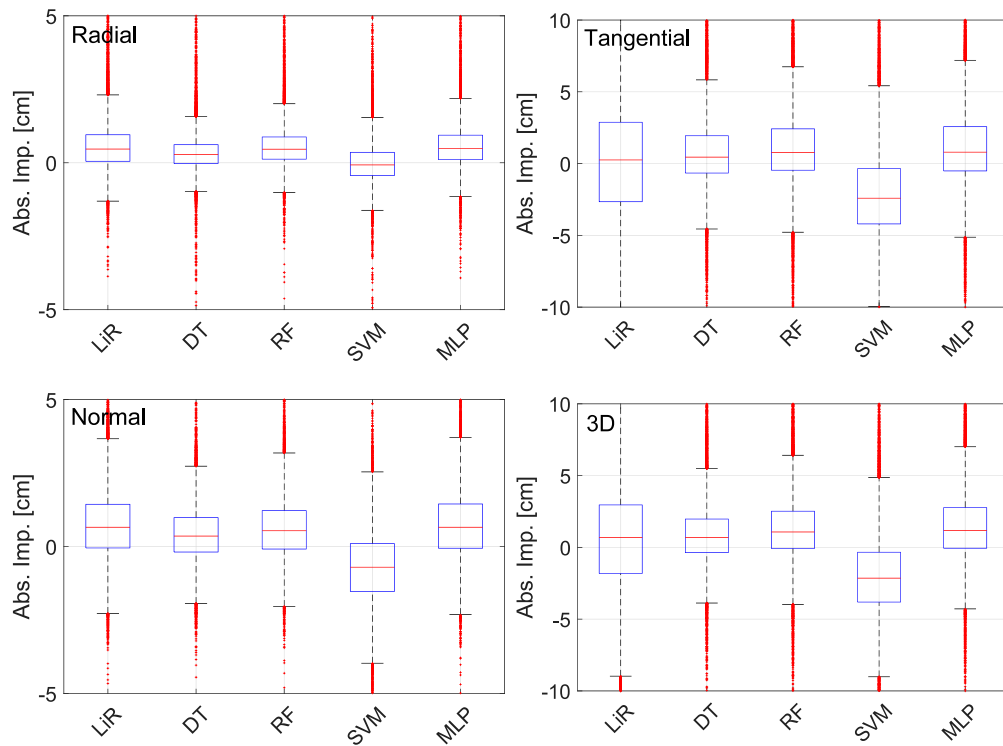


Figure A.1: The version with absolute values in centimeter corresponds to Fig. 4.2

### A.1.2 Deep learning

In this section, we report the absolute improvements (in centimeter) of individual samples in box plots. All the figures have the corresponding relative version in Section 4.2. The layouts are the same as their correspondences.

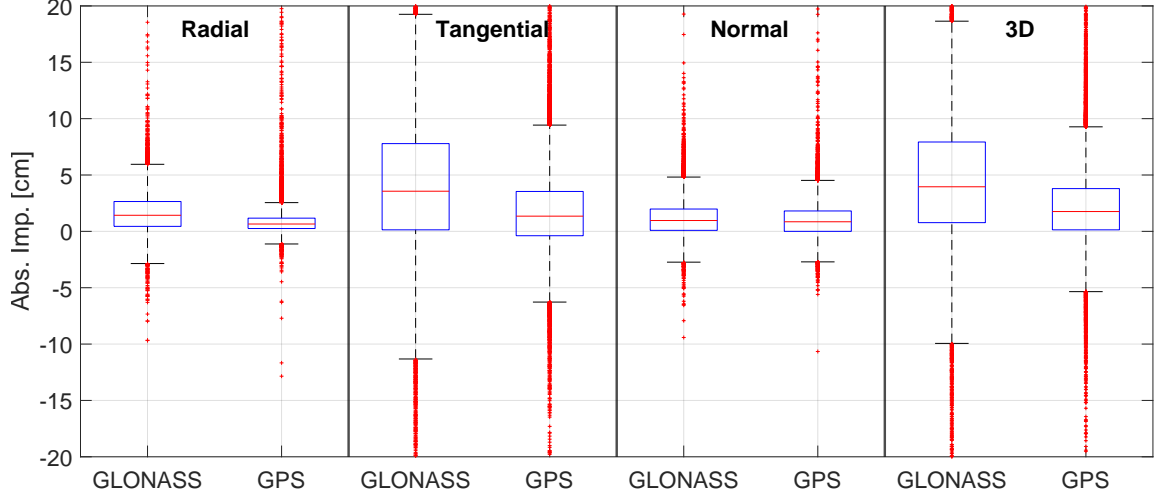


Figure A.2: The version with absolute values in centimeter corresponds to Fig. 4.11

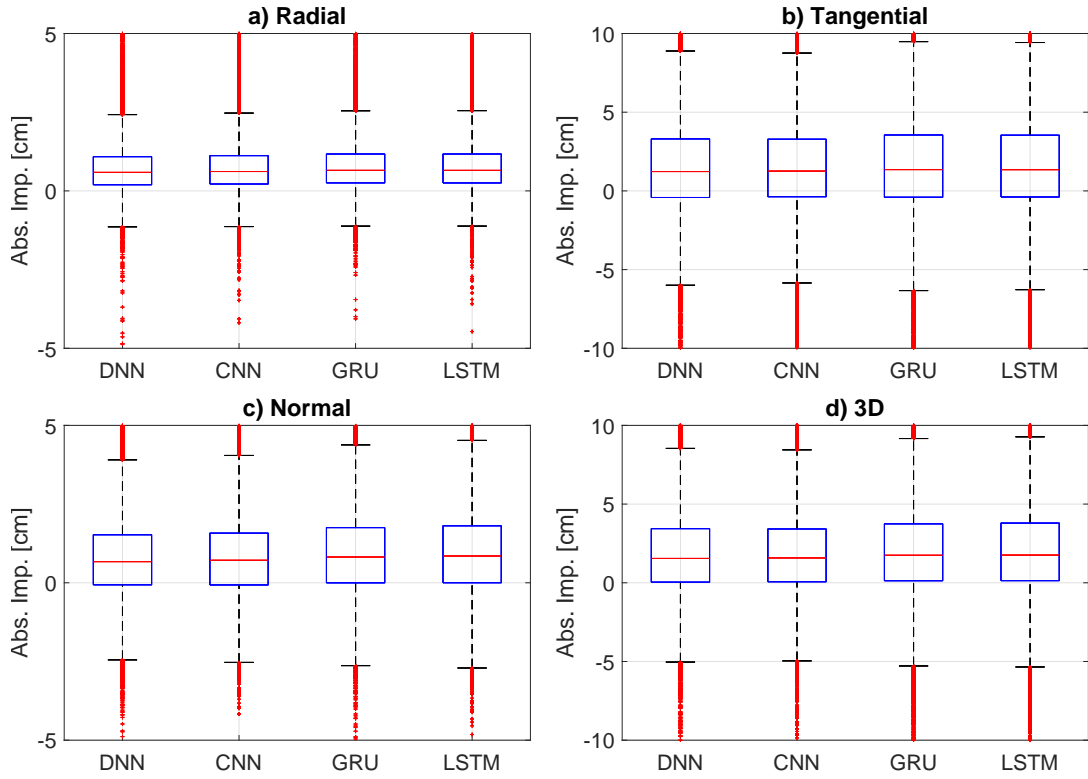


Figure A.3: The version with absolute values in centimeter corresponds to Fig. 4.5

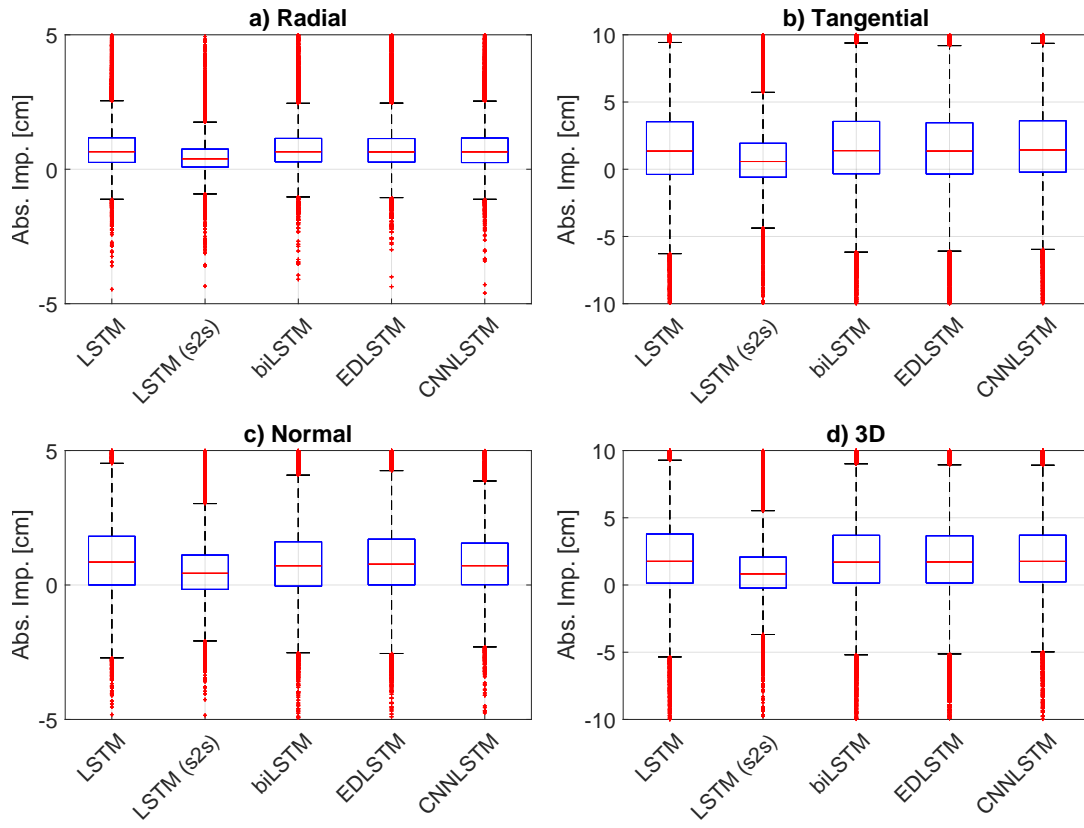


Figure A.4: The version with absolute values in centimeter corresponds to Fig. 4.7

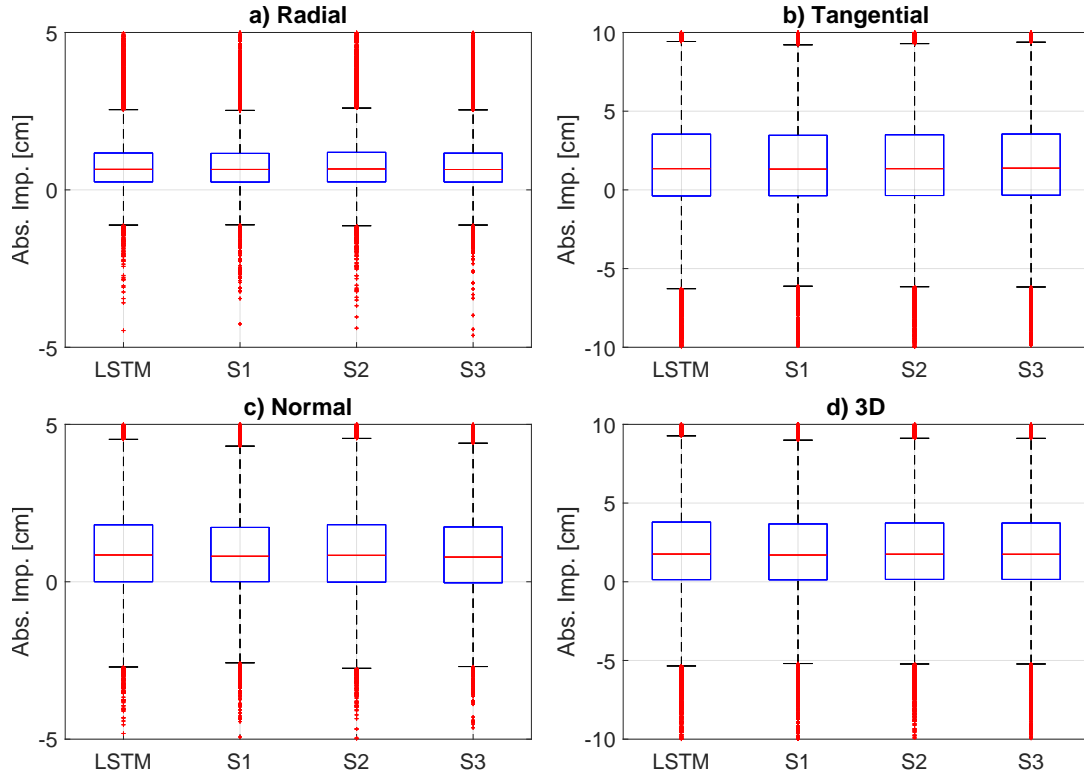


Figure A.5: The version with absolute values in centimeter corresponds to Fig. 4.9

# Appendix B

## Author Contributions

### B.1 Christine Rösch

- Chapter 2 Data
- Section 3.1 Overview
- Section 3.3 Machine Learning Algorithms
- Section 3.5 Feature inclusion and Grouping
- Section 4.1 Machine Learning
- Chapter 5 Conclusion and Outlook

### B.2 Junyang Gou

- Chapter 1 Introduction
- Section 3.1 Overview
- Section 3.2 Data Preprocessing
- Section 3.4 Deep Learning Algorithms
- Section 3.6 Evaluation metrics
- Section 4.2 Deep Learning
- Chapter 5 Conclusion and Outlook

# Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X. (2015), ‘TensorFlow: Large-scale machine learning on heterogeneous systems’. Software available from tensorflow.org.

**URL:** <https://www.tensorflow.org/>

Alpha Magnetic Spectrometer (2021), ‘Thermal Operation’.

**URL:** <https://ams02.space/operations/thermal-operation>

Avinash Navlani (2020), ‘Multi-Layer Perceptron Neural Network using Python’.

**URL:** <https://machinelearninggeek.com/multi-layer-perceptron-neural-network-using-python/>

Bahdanau, D., Cho, K. and Bengio, Y. (2014), ‘Neural machine translation by jointly learning to align and translate’, *arXiv preprint arXiv:1409.0473* .

Bai, S., Kolter, J. Z. and Koltun, V. (2018), ‘An empirical evaluation of generic convolutional and recurrent networks for sequence modeling’, *arXiv preprint arXiv:1803.01271* .

Bengio, Y., Simard, P. and Frasconi, P. (1994), ‘Learning long-term dependencies with gradient descent is difficult’, *IEEE transactions on neural networks* **5**(2), 157–166.

Bento, Carolina (2021), ‘Multilayer Perceptron Explained’.

**URL:** <https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141>

- Bergen, K. J., Johnson, P. A., Maarten, V. and Beroza, G. C. (2019), ‘Machine learning for data-driven discovery in solid Earth geoscience’, *Science* **363**(6433).
- Biewald, L. (2020), ‘Experiment tracking with weights and biases’. Software available from wandb.com.  
**URL:** <https://www.wandb.com/>
- Butt, J., Wieser, A., Gojcic, Z. and Zhou, C. (2021), ‘Machine learning and geodesy: A survey’, *Journal of Applied Geodesy* **15**(2), 117–133.
- Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014), ‘Learning phrase representations using RNN encoder-decoder for statistical machine translation’, *arXiv preprint arXiv:1406.1078*.
- Colah, C. (2015), ‘Understanding LSTM networks’.  
**URL:** <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- Crocetti, L., Schartner, M. and Soja, B. (2021), ‘Discontinuity Detection in GNSS Station Coordinate Time Series Using Machine Learning’, *Remote Sensing* **13**(19), 3906.
- Das, Nipa (2020), ‘Introduction to Random Forest supervised machine learning model’.  
**URL:** <https://blog.probyto.com/introduction-to-random-forest-classifier/>
- Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K. and Darrell, T. (2015), Long-term recurrent convolutional networks for visual recognition and description, in ‘Proceedings of the IEEE conference on computer vision and pattern recognition’, pp. 2625–2634.
- Elliott, Stephen (2000), *Signal processing for active control*, Elsevier.
- Fred Espenak (2021), ‘Geocentric Ephemeris for Sun’.  
**URL:** <http://www.astropixels.com/ephemeris/sun/sun2019.html>
- Goodfellow, I., Bengio, Y. and Courville, A. (2016), *Deep learning*, MIT press.
- Gou, J., Kiani Shahvandi, M., Hohensinn, R. and Soja, B. (2021), Ultra-short-term prediction of LOD using LSTM neural networks, in ‘EGU General Assembly Conference Abstracts’, pp. EGU21–2308.



- Han, S., Qubo, C. and Meng, H. (2012), Parameter selection in SVM with RBF kernel function, in ‘World Automation Congress 2012’, IEEE, pp. 1–4.
- Hochreiter, S. and Schmidhuber, J. (1997), ‘Long short-term memory’, *Neural Computation* **9**(8), 1735–1780.
- IGS (2021), ‘International GNSS Service’.  
**URL:** <https://igs.org/products/>
- Irrgang, C., Saynisch-Wagner, J., Dill, R., Boergens, E. and Thomas, M. (2020), ‘Self-Validating Deep Learning for Recovering Terrestrial Water Storage From Gravity and Altimetry Measurements’, *Geophysical Research Letters* **47**(17), e2020GL089258.
- Jena, M. and Dehuri, S. (2020), ‘Decision tree for classification and regression: A state-of-the art review’, *Informatica* **44**(4), 405–420.
- Kingma, D. P. and Ba, J. (2014), ‘Adam: A method for stochastic optimization’, *arXiv preprint arXiv:1412.6980*.
- Koehrsen, W. (2018), ‘A conceptual explanation of bayesian hyperparameter optimization for machine learning’, *Retrieved July* **11**, 2020.
- Lea, C., Vidal, R., Reiter, A. and Hager, G. D. (2016), Temporal convolutional networks: A unified approach to action segmentation, in ‘European Conference on Computer Vision’, Springer, pp. 47–54.
- LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998), ‘Gradient-based learning applied to document recognition’, *Proceedings of the IEEE* **86**(11), 2278–2324.
- Männel, B., Brandt, A., Nischan, T., Brack, A., Sakic, P. and Bradke, M. (2020a), ‘GFZ final product series for the International GNSS Service (IGS)’.
- Männel, B., Brandt, A., Nischan, T., Brack, A., Sakic, P. and Bradke, M. (2020b), ‘GFZ ultra-rapid product series for the International GNSS Service (IGS)’.
- Matzka, J., Stolle, C., Yamazaki, Y., Bronkalla, O. and Morschhauser, A. (2021), ‘The geomagnetic Kp index and derived indices of geomagnetic activity’, *Space Weather* **19**(5), e2020SW002641.

- Montenbruck, O. and Gill, E. (2012), *Satellite orbits: models, methods and applications*, Springer Science & Business Media.
- Mortlock, T. and Kassas, Z. M. (2021), Assessing machine learning for LEO satellite orbit determination in simultaneous tracking and navigation, *in* ‘2021 IEEE Aerospace Conference (50100)’, IEEE, pp. 1–8.
- NASA Space Place (2021), ‘Sunspots and Solar Flares’.  
**URL:** <https://spaceplace.nasa.gov/solar-activity/en/>
- Pascanu, R., Mikolov, T. and Bengio, Y. (2013), On the difficulty of training recurrent neural networks, *in* ‘International conference on machine learning’, PMLR, pp. 1310–1318.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011), ‘Scikit-learn: Machine learning in Python’, *Journal of Machine Learning Research* **12**, 2825–2830.
- Peng, H. and Bai, X. (2018a), ‘Exploring capability of support vector machine for improving satellite orbit prediction accuracy’, *Journal of Aerospace Information Systems* **15**(6), 366–381.
- Peng, H. and Bai, X. (2018b), ‘Improving orbit prediction accuracy through supervised machine learning’, *Advances in Space Research* **61**(10), 2628–2646.
- Petit, G. and Luzum, B. (2010), IERS conventions (2010), Technical report, Bureau International des Poids et mesures sevres (france).
- Pihlajasalo, J., Leppäkoski, H., Ali-Löytty, S. and Piché, R. (2018), Improvement of GPS and BeiDou extended orbit predictions with CNNs, *in* ‘2018 European Navigation Conference (ENC)’, pp. 54–59.
- Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., Carvalhais, N. et al. (2019), ‘Deep learning and process understanding for data-driven Earth system science’, *Nature* **566**(7743), 195–204.
- Rothacher, M. (2021), ‘Space Geodesy’, *Lecture notes, Institute of Geodesy and Photogrammetry, ETH Zurich*.

- Rothacher, M., Hugentobler, U., Steigenberger, P. and Schmid, R. (2021), ‘Globale Satellitennavigationssysteme’, *Lecture notes, Institute of Geodesy and Photogrammetry, ETH Zurich*.
- Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986), ‘Learning representations by back-propagating errors’, *Nature* **323**(6088), 533–536.
- Ruttner, P., Hohensinn, R., D’Aronco, S., Wegner, J. D. and Soja, B. (2022), ‘Modeling of Residual GNSS Station Motions through Meteorological Data in a Machine Learning Approach’, *Remote Sensing* **14**(1), 17.
- Schuh, H., Ulrich, M., Egger, D., Müller, J. and Schwegmann, W. (2002), ‘Prediction of Earth orientation parameters by artificial neural networks’, *Journal of Geodesy* **76**(5), 247–258.
- Schuster, M. and Paliwal, K. K. (1997), ‘Bidirectional recurrent neural networks’, *IEEE transactions on Signal Processing* **45**(11), 2673–2681.
- Seyoum, W. M., Kwon, D. and Milewski, A. M. (2019), ‘Downscaling GRACE TWSA data into high-resolution groundwater level anomaly using machine learning-based models in a glacial aquifer system’, *Remote Sensing* **11**(7), 824.
- Shahvandi, M. K. and Soja, B. (2021), Modified Deep Transformers for GNSS Time Series Prediction, in ‘2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS’, IEEE, pp. 8313–8316.
- Sharp, Tom (2021), ‘An introduction to Support Vector Regression (SVR)’.  
**URL:** <https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2>
- Space Weather Canada (2021), ‘Solar Flux data’.  
**URL:** <https://www.spaceweather.gc.ca/forecast-prevision/solar-solaire/solarflux/sx-3-en.php>
- Space Weather Prediction Center (2021), ‘Total Electron Content’.  
**URL:** <https://www.swpc.noaa.gov/phenomena/total-electron-content>
- Suthaharan, S. (2016), Support vector machine, in ‘Machine learning models and algorithms for big data classification’, Springer, pp. 207–235.

Sutskever, I., Vinyals, O. and Le, Q. V. (2014), Sequence to sequence learning with neural networks, *in* ‘Advances in neural information processing systems’, pp. 3104–3112.

University of Bern (2021), ‘Ionospheric data’.

**URL:** <http://ftp.aiub.unibe.ch/CODE/2020/>

U.S. Coast Guard Navigation Center (2021), ‘GPS Constellation’.

**URL:** <https://navcen.uscg.gov/?Do=constellationStatus>

Vladimir Lyashenko (2021), ‘How to use random forest for regression’.

**URL:** <https://cnvrg.io/random-forest-regression/>

Yu, S. and Ma, J. (2021), ‘Deep learning for geophysics: Current and future trends’.