Improvement of GPS and BeiDou extended orbit predictions with CNNs

Jaakko Pihlajasalo, Helena Leppäkoski, Simo Ali-Löytty, Robert Piché
Tampere University of Technology, Tampere, Finland
Email: {jaakko.pihlajasalo, helena.leppakoski, simo.ali-loytty, robert.piche}@tut.fi

***Abstract*—This paper presents a method for improving the accuracy of extended GNSS satellite orbit predictions with convolutional neural networks (CNN). Satellite orbit predictions are used in self-assisted GNSS to reduce the Time to First Fix of a satellite positioning device. We describe the models we use to predict the satellite orbit and present the improvement method that uses CNN. The CNN estimates future prediction errors of our model and these estimates are used to correct our orbit predictions. We also describe how the neural network can be implemented into our prediction algorithm. In tests with GPS and BeiDou data, the method significantly improves orbit prediction accuracy. For example, the 68% error quantile of 7 day orbit prediction errors of GPS satellites was reduced by 45% on average.**

I. Introduction

GNSS satellite orbit predictions are a part of self-assisted ephemeris extensions, also called self-assisted GNSS [1]. The goal of self-assisted GNSS is to reduce the Time to First Fix (TTFF), which is the time required for a cold-started positioning device to give an estimate. With self-assisted GNSS the device only needs the signal from the satellite for a position estimate, which reduces TTFF from at least 30 seconds (the time needed for GNSS satellite to broadcast its full navigation message) to about 5 seconds. Broadcast data can be sent to a positioning device using data servers. However, a constant internet connection might be expensive especially for users travelling abroad. One alternative is to compute an estimate of upcoming broadcast data on the device in advance.

Our group has previously presented algorithms for predicting the orbit [2]–[4] and the clock offset [2], [5] related to self-assisted GNSS. We have also presented research on initial state estimation and model parameters in [6]. The present work will focus on orbit prediction.

The orbit of a satellite can be predicted by integrating the equation of motion of the satellite from initial conditions. The satellite's equation of motion is a differential equation that includes forces acting on a satellite. In this work, we only take into account the four largest forces affecting the satellite, which are the gravitations of the Earth, the Sun and the Moon and solar radiation pressure (SRP). Our studies on models of SRP have been presented in [7] and [8]. In [3] we considered adding smaller forces, but found out that this improves the prediction accuracy only slightly. A hybrid model that consists of both mechanistic and data-driven methods was tested in [4]. With the success of deep learning in recent years, we decided to try deep learning methods for our research as well.

In this paper, we consider using convolutional neural networks to improve the orbit prediction accuracy. The CNN is used to estimate the future error of our mechanistic model. This error prediction can then be used to correct our orbit predictions accordingly. The input of the CNN is made from data available from broadcast ephemerides (BE) before the beginning of the prediction.

The rest of this article is organized as follows: in Section II we describe the models we use for GNSS orbit prediction. In Section III we describe the processing steps needed for the prediction. The empirical setup and the accuracy results on the orbit prediction are presented in Section IV and in Section VI we conclude the article.

II. Orbit models and CNN

In this section, we present the models we use in the extended prediction of the satellite orbit. We also describe the convolutional neural network architecture.

A. Extended orbit prediction

We consider the four largest forces affecting the satellite in our force model. Thus, the model for the satellite's equation of motion is given by

$$\ddot{\mathbf{r}}_{\text{Sat}} = \mathbf{a}_{\text{Earth}} + \mathbf{a}_{\text{Sun}} + \mathbf{a}_{\text{Moon}} + \mathbf{a}_{\text{SRP}}\,, \tag{1}$$

where $\mathbf{a}_{\text{Earth}}$, $\mathbf{a}_{\text{Sun}}$ and $\mathbf{a}_{\text{Moon}}$ are accelerations caused by the gravitation of the Earth, the Sun and the Moon. The final term $\mathbf{a}_{\text{SRP}}$ is the acceleration caused by SRP. Our force model is largely based on what is described in [9].

The gravitational potential of the Earth can be modeled with Legendre polynomials as

$$\begin{aligned} U_{\text{E}} = &\frac{GM_{\text{E}}}{r} \sum_{n=0}^{N} \sum_{m=0}^{n} \left[\left(\frac{R_{\text{E}}}{r} \right)^n P_{nm}(\sin\phi) \right. \\ &\left. \Big(C_{nm}\cos(m\lambda) + S_{nm}\sin(m\lambda) \Big) \right], \end{aligned} \tag{2}$$

where r is the distance of the satellite from the Earth's center, M_{E} and R_{E} are the Earth's mass and radius, λ is the longitude and ϕ is the latitude of the satellite. The terms P_{nm} are the associated Legendre polynomials of degree n and order m. The number of terms N used in our numerical model is 8. The values for the coefficients C_{nm} and S_{nm} are from

EGM2008 model [10]. The potential needs to be calculated in ECEF frame and, denoting the transformation matrix from inertial coordinate system to ECEF as R, we may calculate the acceleration caused by Earth in the inertial frame with

$$\mathbf{a}_{\text{Earth}} = \mathrm{R}^{-1}\nabla U_{\mathrm{E}}\,, \tag{3}$$

where ∇ denotes the gradient.

The gravitation of the Sun and the Moon are modeled as point masses. The acceleration caused by a celestial body due to gravitation is

$$\mathbf{a}_{\text{cb}} = GM_{\text{cb}}\left(\frac{\mathbf{r}_{\text{cb}} - \mathbf{r}}{||\mathbf{r}_{\text{cb}} - \mathbf{r}||^3} - \frac{\mathbf{r}_{\text{cb}}}{||\mathbf{r}_{\text{cb}}||^3}\right), \tag{4}$$

where M_{cb} is the mass of the body, $\mathbf{r}_{\text{cb}}$ is its position in an Earth-centered inertial frame, and $\mathbf{r}$ is the position of the satellite in the same frame.

For solar radiation pressure (SRP), we use a two-parameter empirical model presented in [7]. The acceleration caused by SRP is presented as a sum of two directional components. The acceleration caused by SRP is

$$\mathbf{a}_{\text{SRP}} = \nu\big(-\alpha_1\frac{\mathbf{r}_{\text{S}}}{||\mathbf{r}_{\text{S}}||^3} + \alpha_2\mathbf{e}_{\text{y}}\big), \tag{5}$$

where α_1 and α_2 are parameters that we estimate by fitting our force model to the precise orbit data [7]. Vector $\mathbf{r}_{\text{S}}$ points from the satellite to the Sun, i.e., $\mathbf{r}_{\text{S}} = \mathbf{r}_{\text{Sun}} - \mathbf{r}_{\text{Sat}}$, and $\mathbf{e}_{\text{y}}$ is the unit vector in the satellite's y-direction (parallel to the solar panel axis), which is defined as

$$\mathbf{e}_{\text{y}} = \frac{\mathbf{r}_{\text{Sat}} \times \mathbf{r}_{\text{S}}}{||\mathbf{r}_{\text{Sat}} \times \mathbf{r}_{\text{S}}||}\,, \tag{6}$$

where $\times$ denotes the vector product. Parameter $\nu \in [0, 1]$ in 5 is a factor that accounts for the Earth's shadow, with $\nu = 1$ when the satellite is not in Earth's shadow and $\nu = 0$ when it is fully in Earth's shadow. We use the conical model for Earth's shadow described in [9].

B. CNN architecture

In this section, we describe the general architecture and design of the CNN we use. Since the focus of this work is not the neural network itself, we will only present the general principles of the CNNs. Architecture and design of a CNN consist of different layers and training with backpropagation. For our network, we use input image, convolution, rectified linear unit and fully connected layers.

TABLE I: Layer and training parameters of the network used.

Convolutional layer	
Number of filters	8
Filter size	Width of image
Padding	0
Stride	1
Training parameters	
Learn rate	0.001
Momentum rate	0.9
Number of training epochs	30

The network and its individual layers can be described as a function which gives an output for some input. The input image size I_s is $width \times height \times depth$ with depth being the number of color channels. The input image layer zero-centers and normalizes the data for each color channel.

2-D convolution is computed for all channels of the image in convolutional layers. The convolution filter B with trainable weights is slid across the image A according to the chosen padding and stride. Padding means adding zeros to the edges of the image and stride means the step size of the filter. The output of a convolutional filter is defined as a sum of Hadamard product matrix elements, which can be presented in matrix multiplication as

$$C_{(i,j)} = \frac{1}{dn}\sum_{k=1}^{d}\text{tr}(\mathrm{A}_k\mathrm{B}_k^T) + b, \tag{7}$$

where $C_{(i,j)}$ is the (i, j)-th element of the convolution matrix C, n is the number of elements in one channel of the image, d represents the number of color channels and b is bias of the filter. A_i and B_i are the matrices of color channel i. When working with images and filters of different sizes, the filter B is slid across matrix A and the outputs $C_{(i,j)}$ form a matrix C with elements according to the stride of the filter. [11]

Since convolution is a linear operation a non-linearity needs to be added to the network to model nonlinear variations of response variables. Non-linearity can be handled by using rectified linear unit (ReLU), which is defined as

$$f(x) = \max(0, x), \tag{8}$$

where x is the element of the image to be non-linearized. [12]

Finally all the neurons of the network are connected in a fully connected layer. Outputs of all the previous layers are connected and used to form the final output of the network. The outputs of the previous layer are vectorized and then the output of the fully connected layer is computed as matrix multiplication. The output $\mathbf{a}_{\text{fcl}}$ of fully connected layer is

$$\mathbf{a}_{\text{fcl}} = \mathrm{W}\begin{bmatrix}\mathbf{a}_{\text{fcl,1}}\\ \mathbf{a}_{\text{fcl,2}}\\ \vdots\\ \mathbf{a}_{\text{fcl,n}}\end{bmatrix} + \mathbf{b}_{\text{fcl}} = \mathrm{W}\mathbf{a}_{\text{input}} + \mathbf{b}_{\text{fcl}}, \tag{9}$$

where W is the weight matrix of the layer, $\mathbf{a}_{\text{fcl,i}}$ is the output i of previous layer, $\mathbf{b}_{\text{fcl}}$ is the bias vector of the layer and $\mathbf{a}_{\text{input}}$ are the vectorized outputs of the previous layer.

Training of the weights and biases in the network are computed with backpropagation using stochastic gradient descent with momentum (SGDM). Images are propagated through the network layers and the resulting output vector is compared with response values with the loss function. Since we are using the network for regression, the loss function is chosen to be mean squared error (MSE). MSE is defined as

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(\hat{\mathbf{y}}_i - \mathbf{y}_i)^2, \tag{10}$$

where θ are all weights of the network, n is the number of predictions, $\hat{\mathbf{y}}_i$ is the i-th prediction and $\mathbf{y}_i$ is i-th response. The errors are then used to improve the weights. We use SGDM to iterate our weights. SGDM tries to minimize the error using gradient descent, which means taking steps towards the gradient minimum. This is where we use the learn rate η and momentum rate α of the network. Weight update is given by

$$\mathbf{m} = \eta\mathbf{m} + \alpha\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \tag{11}$$

and

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \mathbf{m}, \tag{12}$$

where $\mathbf{m}$ is the momentum vector and $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is the gradient of the loss function. This stepping is continued until a chosen number of gradient descent steps are completed.

III. Prediction of RTN errors with CNN

In this section, we describe our method of using a CNN to improve our satellite orbit predictions from the existing model. We do our error analysis in RTN frame. RTN stands for Radial, Tangential and Normal coordinate system. CNN is trained to learn the RTN errors of an orbit prediction for two weeks from RTN error data from a few days.

A. Error prediction

The RTN errors of orbit predictions from our existing model are treated as the response "images" by the CNN. To obtain the input images, we create data by predicting the orbit backwards in time. The backwards prediction is based on the last BE of the prediction interval. The errors are computed by comparing the predicted positions with the satellite positions based on all the BEs from the prediction interval. We choose to predict two weeks of RTN errors from 4 days of RTN error data with the network. Four days of prediction data with 15 minute time intervals gives us an RTN error matrix of size 3×384 which is reshaped to an RGB images with the width of the image representing the number of orbital periods of the satellite. Each image channel represents one of the RTN directions. The image size is chosen such that one column represents the errors of one orbital period of the satellite. An example of the image representation can be found in Figure 1

The RTN error data needs to be preprocessed for the network. The steps of preprocessing are:

1) Computation of tangential difference
2) Normalization of data
3) Reshaping to image form.

The image data is preprocessed by normalizing all error values to a range from 0 to 1, because RTN errors have different scale and bias. The normalization parameters are chosen to be the maximum and minimum values of all the training data, since the real scale of the RTN errors is not known beforehand. Normalization is done for each individual data point in the input image and the response.

Since the tangential error curve shows linear behavior and isn't zero-centered, we choose to make the curves similar to radial and normal curves by taking the differences of successive points. In Figure 2 is an example of the tangential difference's effect. Taking the difference leaves us with one less data point, so we leave the last radial and normal error from the initial state unused. The tangential difference ΔT is defined by

$$\Delta T_i = T_i - T_{i-1}, \quad i = 2, 3, 4, ..., n, \tag{13}$$

where ΔT_i is the ith tangential difference, T_i is the ith tangential error and n is the length of the error vector. The tangential difference can also be inversed when the tangential error of the initial state is kept for later. The tangential difference is also

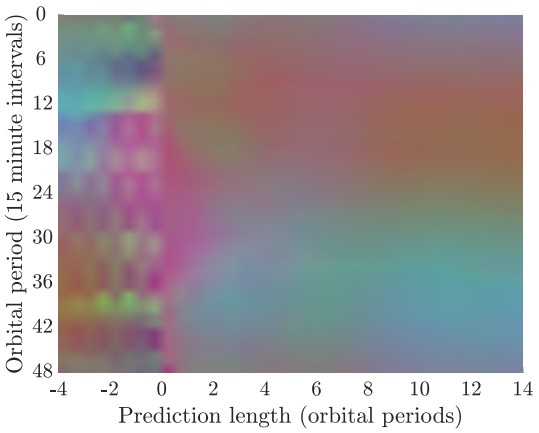

Fig. 1: Image representation of RTN errors of one orbit prediction.

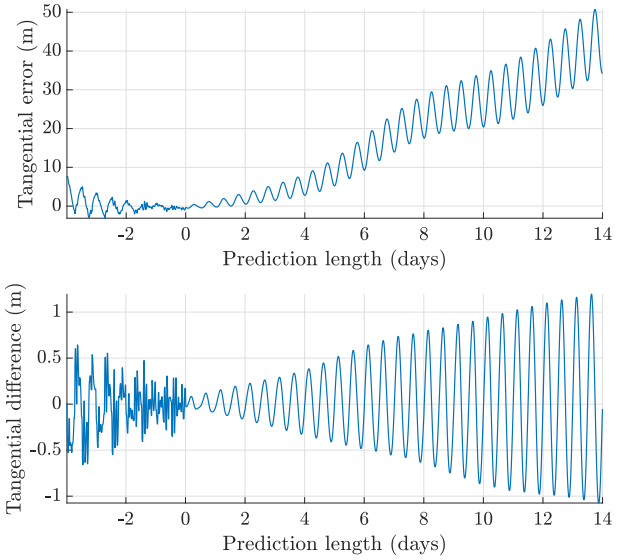

Fig. 2: An example of what tangential difference does to the tangential error curve.

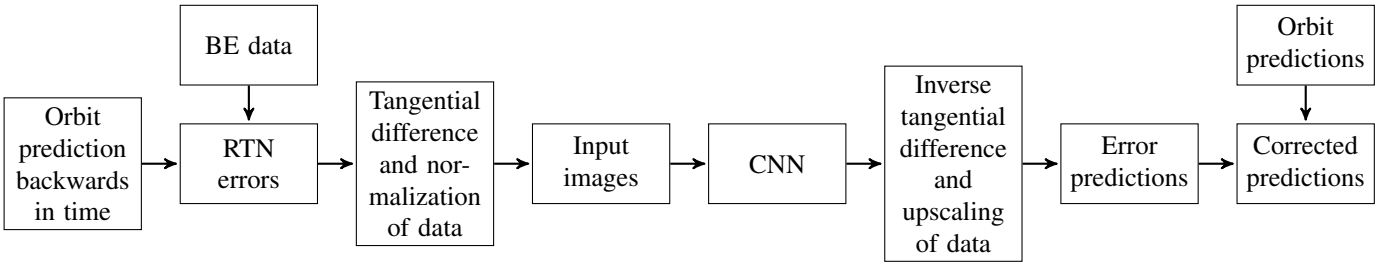

Fig. 3: Block diagram of the process of the CNN method

normalized to the range 0 to 1. Tangential difference estimates from the network can be inversed recursively with

$$\hat{T}_1 = \Delta\hat{T}_1 - T_0 \tag{14}$$

and

$$\hat{T}_i = \Delta\hat{T}_i - \hat{T}_{i-1}, \quad i = 2, 3, 4, ..., n, \tag{15}$$

where $\hat{T}_i$ is the ith tangential error estimate, $\Delta\hat{T}_i$ is the ith tangential difference estimate and T_0 is the tangential error of the initial state.

In Figure 1 is an example of a preprocessed input image. Each pixel represents RTN errors of the prediction in 15 minute time intervals. Orbital periods -4 to 0 are the input of the network and rest is the output response. The output is given as a vector from the network. We present the output in image form to show the periodicity of the orbital periods.

With the preprocessed data, we can train the network to learn the upcoming RTN errors. The CNN was chosen to have a simple architecture, with only a few layers, to work with the small size of the input images. We chose the network to have one convolution layer followed by a rectified linear unit and fully connected layer. The outputs of the network are scaled back with the global normalization parameters from the training data and the tangential difference is inverted with the initial state. The parameters used in our network are presented in Table I.

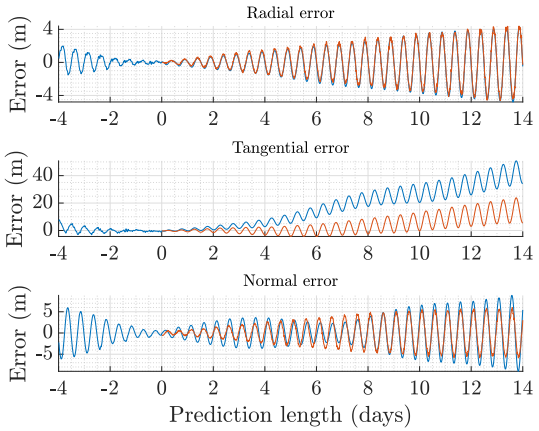

Fig. 4: Example orbit and error predictions of a GPS satellite.

In Figure 4 is an example orbit prediction in blue and corresponding error predictions in red. The negative time represents the input data of the network.

B. Error correction of satellite predictions

The error predictions can be used to correct our orbit predictions. The orbit of the satellite is predicted as before, but we also predict the orbit forwards in time.

As previously described, the CNN gives us the scaled error predictions which are processed to give the error prediction vector $\hat{\mathbf{y}}_{\text{RTN}}$. The steps of estimate processing are:

1) Inverting tangential difference
2) Upscaling with parameters from training data.

To correct the orbit prediction from the existing model we need to change the coordinate system of the error prediction to an inertial coordinate system. This can be done with

$$\hat{\mathbf{y}}_{\text{ECI}} = \begin{bmatrix} \mathbf{e}_{\text{R}}^T & \mathbf{e}_{\text{T}}^T & \mathbf{e}_{\text{N}}^T \end{bmatrix}^{-1} \hat{\mathbf{y}}_{\text{RTN}} = \text{R}_{\text{RTN}}^{-1}\hat{\mathbf{y}}_{\text{RTN}}, \tag{16}$$

with unit vectors

$$\mathbf{e}_{\text{R}} = \frac{\mathbf{r}}{\|\mathbf{r}\|}, \quad \mathbf{e}_{\text{N}} = \frac{\mathbf{r}\times\mathbf{v}}{\|\mathbf{r}\times\mathbf{v}\|} \quad \text{and} \quad \mathbf{e}_{\text{T}} = \mathbf{e}_{\text{R}} \times \mathbf{e}_{\text{N}}, \tag{17}$$

where $\hat{\mathbf{y}}_{\text{ECI}}$ is the error predictions in the inertial frame, $\mathbf{r}$ and $\mathbf{v}$ are the predicted satellite position and velocity respectively. R_{RTN} is the transformation matrix from inertial frame to RTN frame. The error predictions can now be subtracted from the orbit predictions to give the corrected orbit predictions.

IV. Results

We predict the satellite orbits using broadcast ephemerides acquired from IGS [13] for GPS and MGEX [14] for BeiDou. The accuracy of the orbit predictions is assessed by comparing the predictions with precise ephemerides, also obtained from IGS.

The CNN was trained and tested using predictions made during GPS weeks 1835-1870 with around 3000 predictions per satellite. Predictions that had missing data and predictions with errors higher than one kilometer were removed from the training to make the network more reliable. The network was then verified by using it to correct predictions made during weeks 1887-1907.

The results are compared with orbit predictions without the CNN based correction and with LFM, a data-driven orbit-prediction method described in [4]. A single CNN was trained

TABLE II: SISRE weight values for different constellations. Values are from [16].

	GPS	BDS GEO/IGSO	BDS MEO
w_R	0.98	0.99	0.98
$w_{T,N}^2$	$\frac{1}{49}$	$\frac{1}{126}$	$\frac{1}{54}$

to be used with all GPS satellites and one CNN was trained for each orbit type of BeiDou.

The orbital error results are typically represented in the RTN coordinate system, because different directions contribute positioning error different ways. The effects of the RTN errors can be combined to Signal In Space Range Error (SISRE). We use the interpretation of SISRE used in Global Positioning System Standard Positioning Service Performance Standard [15]. Usually, SISRE also includes the errors due to satellite clock, but this paper focuses on orbit-only SISRE, which is calculated with

$$\text{SISRE} = \sqrt{w_R^2 \Delta R^2 + w_{T,N}^2 (\Delta T^2 + \Delta N^2)}, \qquad (18)$$

where ΔR,ΔT and ΔN are error components in radial, transverse and tangential directions, respectively. w_R and $w_{T,N}$ are weight parameters that depend on the satellite constellation. These weights are listed in Table II.

Since SISRE is a non-negative metric we can use quantiles to represent the results. For both constellations, the 68% and 95% quantiles of SISRE are shown as a function of prediction length.

The complete results of the CNN method for all individual GPS and BeiDou satellites and more detailed information about the CNN method can be found in [17]. There the results for both constellation are shown for both 7 and 14 day predictions for 68% and 95% quantiles.

A. GPS

With GPS satellites, the orbit predictions were made every two hours to match the broadcast ephemerides available from IGS. The general results of the method can be seen in Table III and general behavior of SISRE as a function of time in Figure 5. The CNN prediction method improves orbit accuracy for all GPS satellites. The results vary by satellite, but the CNN method also compares well against LFM.

Generally the improvements of 68% quantile for one-week predictions were 30-50% for GPS. This improvement is large in comparison with the improvements made by adding physics-based components to the force model [3], which only improved accuracy by a few percent. The LFM improved around 20-30%, so CNN does better than LFM. The average accuracy and improvements can be viewed from Table III.

B. BeiDou

For BeiDou satellites, we made predictions once every hour for all three orbit types. We present the results for each orbit type separately in Table III. Orbit prediction accuracy for all BeiDou satellites of all orbit types improved greatly.

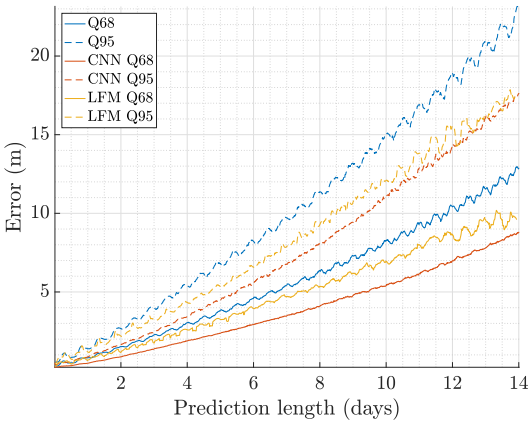

Fig. 5: SISRE quantile curves for GPS PRN 15, with CNN method, with LFM and with neither

The improvements on GEO and IGSO satellites are excellent, since for most of these satellites the 95% quantile is better than the 68% quantile without CNN. Generally GEO satellites improved one-week predictions around 60% and IGSO satellites improved around 50%. The CNN method for GEO and IGSO satellites also works better than LFM.

The improvements of MEO satellites are large with around 70% improvements on one week predictions. The improvement is about the same as what was obtained with LFM. Also the 95% quantile for MEO satellites didn't improve as much with around 40% improvements.

V. Conclusions

In this paper, we presented a method of using convolutional neural networks with an existing model to improve the orbit predictions of GNSS satellites. Our results show that CNNs improve the performance of the mechanistic model. The input of the CNN relies only on previous broadcasts and the usage in the hybrid prediction algorithm is simple. The CNN we used was simple but the method improves accuracy significantly and thus a more complex and deeper network might provide even better results. We also present results against another hybrid model we have previously used. The method improves accuracy significantly for all GPS and BeiDou satellites of all orbit types, but further improvements could be done by training networks for single satellites. Further research is needed for inclusion of other GNSS constellations.

TABLE III: Collection of average results for 7 day predictions

	68% quantile		95% quantile	
Constellation	SISRE (m)	Improvement	SISRE (m)	Improvement
GPS	3.56	44.9%	7.73	42.6%
BeiDou GEO	8.34	60.0%	15.6	53.2%
BeiDou IGSO	6.60	49.9%	11.8	51.7%
BeiDou MEO	4.56	69.7%	16.8	42.4%

ACKNOWLEDGMENT

The authors thank HERE Global B.V. for financial support, guidance and collaboration during this research.

REFERENCES

[1] Ivan G. Petrovski, *GPS, GLONASS, Galileo, and BeiDou for Mobile Devices - From Instant to Precise Positioning*. iP-Solutions, 2014.

[2] M. Seppänen, J. Ala-Luhtala, R. Piché, S. Martikainen, and S. Ali-Löytty, "Autonomous prediction of GPS and GLONASS satellite orbits," *NAVIGATION*, vol. 59, no. 2, pp. 119–134, 2012.

[3] A. Pukkila, J. Ala-Luhtala, R. Piché, and S. Ali-Löytty, "GNSS orbit prediction with enhanced force model," in *Localization and GNSS (ICL-GNSS), 2015 International Conference on*. Gothenburg, Sweden: IEEE, 2015, pp. 1–6.

[4] S. Rautalin, S. Ali-Löytty, and R. Piché, "Latent force models in autonomous GNSS satellite orbit prediction," in *2017 International Conference on Localization and GNSS*, 2017.

[5] S. Martikainen, R. Piché, and S. Ali-Löytty, "Outlier-robust estimation of GPS satellite clock offsets," in *Localization and GNSS (ICL-GNSS), 2012 International Conference on*, Starnberg, Germany, June 2012, pp. 1–5.

[6] J. Ala-Luhtala, M. Seppänen, S. Ali-Löytty, R. Piché, and H. Nurminen, "Estimation of initial state and model parameters for autonomous GNSS orbit prediction estimation of initial state and model parameters for autonomous GNSS orbit prediction," in *International Global Navigation Satellite Systems Society Symposium 2013 (IGNSS2013)*, Gold Coast, Queensland, Australia, July 2013.

[7] J. Ala-Luhtala, M. Seppänen, and R. Piché, "An empirical solar radiation pressure model for autonomous GNSS orbit prediction," in *Proceedings of PLANS 2012 IEEE/ION Position Location and Navigation Symposium*, April 2012, pp. 568–575.

[8] H. Leppäkoski, S. Rautalin, X. Zhang, S. Ali-Löytty, and R. Piché, "Extended prediction of QZSS orbit and clock," in *2016 International Conference on Localization and GNSS*, 2016.

[9] O. Montenbruck and E. Gill, *Satellite Orbits: Models, Methods and Applications*, 3rd ed. Springer, 2005.

[10] National Geospatial-Intelligence Agency. EGM2008 model coeffcients. [Online]. Available: http://earth-info.nga.mil/GandG/wgs84/gravitymod/egm2008/first_release.html

[11] B. Rohrer, "How do Convolutional Neural Networks work?" Aug. 2016. [Online]. Available: https://brohrer.github.io/how_convolutional_neural_networks_work.html

[12] R. Ng, "Deep Convolutional Networks," 2017. [Online]. Available: "http://www.ritchieng.com/machine-learning/deep-learning/convs/"

[13] "IGS Products." [Online]. Available: http://www.igs.org/products

[14] "MGEX products." [Online]. Available: http://mgex.igs.org/IGS_MGEX_Products.html

[15] "Global Positioning System Standard Positioning Service Performance Standard," Tech. Rep., Sep. 2008.

[16] O. Montenbruck, P. Steigenberger, and A. Hauschild, "Broadcast versus precise ephemerides: a multi-GNSS perspective," *GPS solutions*, pp. 321–333, 2015.

[17] J. Pihlajasalo, "Improvement of satellite orbit prediction accuracy and quality with deep learning and spectral analysis," Master's thesis, Tampere University of Technology, Nov. 2017. [Online]. Available: http://URN.fi/URN:NBN:fi:tty-201710262067