Using real-time deep learning algorithms to predict optimal traffic flow during bridge opening times

Daniël R.L. Overdevest[1,2], Michael R. de Winter [1], Dennis van Muijen [1,3] and Joana F.M.F. Cardoso[1*]

Abstract
The province of Zuid-Holland operates and maintains more than 100 bridges and has, therefore, an important role in controlling traffic flow in the province. Smart IT solutions offer the possibility to combine real-time road and vessel traffic information using sensors and algorithms. The project 'Impactmonitor Brugopeningen' aims at predicting traffic intensity around several important bridges in the province of Zuid-Holland. Long Short-Term Memory (LSTM) neural networks were used to predict road traffic intensity up to 21 minutes ahead, resulting in 75-95% accuracy. Models were deployed in the cloud using Microsoft Azure and delivered as a user-friendly web application where real-time road and vessel traffic information is presented. This information is used to advise bridge operators on the optimal time window to open a bridge. In the future, traffic systems, such as navigation software, may use this information to consider planned bridge opening periods. By using the web application, bridge operators can combine their practical knowledge with decision supporting software based on recent AI technology.

Keywords
Bridge monitoring system (BMS), traffic flow, deep learning, Long Short-Term Memory (LSTM), traffic congestion model, data science, prediction model, real-time, API

[1] *Province of Zuid-Holland, Programme 'Transparante Open Provincie (TOP) - Opgave Vernieuwing Datawarehouse', Department I & A, Zuid-Hollandplein 1, 2509 LP Den Haag*
[2] *DO IT analytics: daniel@doit-analytics.nl*
[3] *Affine Solutions: d@affine.nl*
[*] *Correspondentie: j.cardoso@pzh.nl*

Contents

1. Introduction

In the water-rich and densely populated province of Zuid-Holland, a smooth road and vessel traffic flow is essential. The province is responsible for operating and maintaining more than 100 bridges and fifty-five of these have sensors that monitor if the bridge is open or closed. For an increasing number of bridges, it is a challenge to optimize road traffic flow, considering vessel traffic and the increasing road traffic intensity. The province is committed to invest in a smart, safe and efficient traffic flow to tackle this challenge. This is done through collaborative projects with regional partners, such as the projects Bereik! [1], Blauwe Golf Verbindend (Blue Wave Connects) [2] and the Smart Shipping programme [3].

The departments of Infrastructure Management Services (DBI) and Information & Automation (I&A) of the province of Zuid-Holland have been working on an innovative IT solution: the 'Impactmonitor Brugopeningen'. This application monitors the effect of bridge openings on road traffic flow. It gives bridge operators real-time information on the optimal

time window to open a bridge. This is accessed by analysing the impact of a bridge opening in terms of traffic flow and CO_2 emissions. Although freight transport via waterways is considered as a more sustainable transport option, a bridge opening causes traffic jams and queues and has, therefore, a negative environmental impact. This is especially important during rush hours since opening a bridge 5-15 minutes earlier or later can have considerable impact on the total vehicle loss hours. The web-application described here combines 21-minute ahead predictions using deep neural networks to predict traffic intensity combined with real-time information on vessel traffic. As a starting point, three bridges were analysed and included in the application: the Coenecoopbrug (Waddinxveen), the Lammebrug (Leiden) and the Kruithuisbrug (Delft). The application is a decision supporting tool, adjusted to the behaviour and workflow of bridge operators. The goal is to show not only real-time traffic information but also to advise bridge operators on the optimal time window to open a bridge, in order to minimise traffic congestion and the CO_2 footprint. The application is currently being tested by a group of bridge operators, after which it will be evaluated and gradually introduced into bridge control centres.

At the technical level, recent developments in big data, AI and application development were used (Figure 1): real-time data on traffic intensity was collected from road sensors and analysed with neural networks; the expected traffic intensity was predicted; and data was obtained from several API's. The result is a modern web application. The present paper describes not only the technical aspects of data analysis but also the used approach and machine learning tools, and, ultimately, the resulting user-friendly interface.

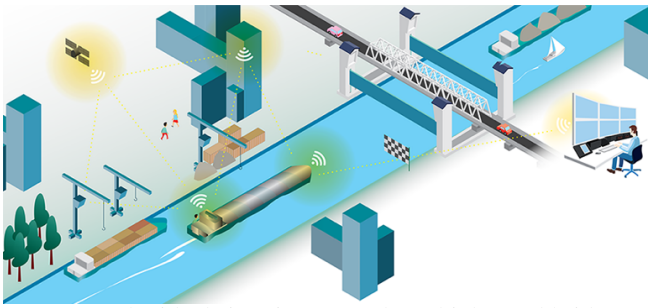

Figure 1. The ideal situation: vessels, vehicles and bridge operators are connected to each other via smart sensors.

2. Data preparation

2.1 Datasets

HIG Traffic Systems road sensors In many provincial roads, sensors placed in the road measure traffic flow in terms of number of vehicles, vehicle type and speed. However, only a small part of these sensors is connected real-time to the National Data Warehouse for Traffic Information (NDW) while the rest is used for monitoring and evaluation. Data from sensors relevant for this project which were not available through NDW, were obtained directly from HIG Traffic Systems B.V. for the three bridges mentioned in the introduction. In order to further use this data, individual measurements were combined for each sensor to determine the total number of vehicles, the average speed and length, and the number of vehicles per vehicle class. Between one and two years of historical data was used to train the model, depending on data availability.

NDW real-time road traffic intensity The database of NDW has real-time data on the number of vehicles per minute per sensor. This real-time data is used to real-time traffic intensity predictions.

COBALD bridge opening data COBALD is a system developed by the Province of Zuid-Holland which is connected to bridge control centres. All bridge openings (beginning and end time) are registered in this system. This historical dataset was used to prepare and analyse road sensor data.

Bridge Sense real-time data Extra sensors are placed on most bridges for measuring real-time bridge opening periods. This data is passed to the 'Blue Wave Connects' portal and to NDW, which make the data available as Open Data. This real-time data was used to determine the bridge status (open vs. closed) in the web application developed here.

2.2 Data quality and cleaning

Since sensors may have measuring faults, several quality controls and cleaning actions were needed before data could be used for analysis. Data cleaning was based on descriptive statistics and outliers:

- Sensor: Vehicles longer than 18.7 meters were discarded.
- Sensor: Vehicle speed higher than 120 km/h were discarded.
- Bridge opening: Bridge openings shorter than 3 minutes and longer that 14 minutes were discarded.

Traffic intensity during road traffic constraints (such as bridge openings) may be misleading, since during a bridge opening the sensor does not measure any passing vehicle. Therefore, traffic intensity during bridge openings and in a 5-minute range around these openings were corrected with the last measured intensity using a forward fill.

2.3 Used tools and techniques

Prediction model development Deep learning models were developed in the Keras library (version 2.2.4), an open source package used for modelling neural networks in Python [4]. Keras is designed to enable fast experimentation with deep neural networks [5]. Keras was used with the TensorFlow back-end (version 1.5.0) [6], whereby in a first experimental phase, the GPU variant of the chosen model (section 3) was used. The grid search capability from the Scikit-learn machine learning library for the Python programming language was used to tune the hyper-parameters [7]. All analyses related to

data cleaning and modelling traffic intensity were performed in Python.

Model evaluation During training and model selection, several statistical metrics were used to quantify the accuracy of the models in predicting historical data which had not been presented to the model before. In this way, the model can be tested on data that is similar to real-time data. The following metrics were used:

- R-squared (R^2) - the squared correlation between the observed values and the predicted values by the model. The higher the adjusted R^2, the better the model.
- Mean Squared Error (MSE) - the average squared difference between the observed values and the values predicted by the model. The lower the MSE, the better the model.
- Mean Absolute Percentage Error (MAPE) – the average of the absolute percentage difference between observed and predicted values. The smaller the MAPE, the better the model.

To evaluate the performance of trained models, it is important to test them on unseen data. Cross-validation (CV) is one of the techniques used to test the effectiveness of machine learning models. To perform CV, a portion of the data is not used to train the model but is kept aside and used later to test the model. In the present work, the train-test split approach was used whereby 75% of the data was used to train the models (train set) and 25% to test (test set). The metrics described above were used to measure model performance during CV.

3. Modelling and predicting traffic intensity

3.1 LSTM neural networks

Artificial Neural Networks (ANN) are a set of algorithms, inspired by the highly efficient way the human brain works, which are designed to recognize patterns [8][9][10]. An ANN consists of a collection of artificial neurons organized in three interconnected layers: input, hidden (that may include more than one layer) and output. Due to recent developments in the field of machine learning and increase in computing power, these networks can perform complex tasks using deep learning algorithms [11]. Recurrent Neural Networks (RNN) use the output of a layer and feed it back to the input to help predicting the outcome of the layer [12] and are, therefore, useful for time-series analysis. Long Short-Term Memory (LSTM) is a specific and powerful RNN designed to model long temporal sequences[13][14][15]. In this study, LSTM was chosen for predicting real-time traffic intensity for three bridges.

3.2 Selected models

A univariate LSTM model with multi-step forecasting was trained separately for each station of each bridge in Keras, following the network architecture of Brownlee [16]. Two different LSTM models were tested and compared using the metrics mentioned in section 2.3. Initially, a simple LSTM model that reads in a time sequence for traffic intensity data and predicts a vector output of the next sequence was tested (Script 1).

Later, an encoder-decoder LSTM model (Script 2) was also tested, whereby the encoder reads and encodes the input sequence, and the decoder reads the encoded input sequence and makes a one-step prediction for each element in the output sequence [16]. Both approaches predict a sequence output but in the latter, an LSTM model is used in the decoder, allowing it to both know what was predicted for the prior time sequence and accumulate internal state while outputting the sequence.

Script 1. Python code used to train the vector output LSTM model.

```
model = Sequential()
model.add(LSTM( n_layers , activation=activation ,
        kernel_initializer=kernel_initializer ,
        input_shape=(n_steps_in , features )))
model.add(Dense(n_steps_out))

optimizer = optimizer(  lr=lr , beta_1=beta_1 ,
                beta_2=beta_2 ,
                decay=decay , epsilon=epsilon)
model.compile(optimizer=optimizer , loss=loss ,
        metrics=['accuracy', 'mae'])
```

For both the vector output model and the encoder-decoder model, traffic intensity for 21 minutes was used as input sequence (*n_steps_in*) for predicting the traffic intensity in the following 21 minutes (output sequence, *n_steps_out*). Since univariate models were used the number of features was always 1.

Script 2. Python code used to train the encoder-decoder LSTM model.

```
model = Sequential()
model.add(LSTM( n_layers , activation=activation ,
        kernel_initializer=kernel_initializer ,
        input_shape=(n_steps_in , features )))
model.add(RepeatVector(n_steps_out))
model.add(LSTM( n_layers , activation=activation ,
                return_sequences=True))
model.add(TimeDistributed(Dense(dense2 ,
        activation=activation2 ,
        kernel_initializer=kernel_initializer )))
model.add(TimeDistributed(Dense(dense )))

optimizer = optimizer(  lr=lr ,
        beta_1=beta_1 , beta_2=beta_2 ,
        decay=decay , epsilon=epsilon)
model.compile(optimizer=optimizer , loss=loss ,
        metrics=['accuracy','mae'])
```

3.3 Parameter optimization

Hyperparameter optimization is an important step in deep learning model development. Neural networks may be complex and difficult to configure due to the many parameters that need to be optimized. This technique is provided in the GridSearchCV class [13] (see section 2.3).

Table 1. Selected hyper-parameters.

Parameter	Value
epochs	30
batch size	40
epochs	30
batch size	40
n_layers	85
optimizer	Adam
learning_rate (lr)	0.01
beta_1	0.9
beta_2	0.9
decay	0
epsilon	0.7
activation	sigmoid
activation2	hard_sigmoid
kernel_initializer	glorot_uniform
loss	mse
dense1	1
dense2	1024

Hyperparameter optimization was done for the Coenecoopbrug data (Table 1). Selected hyperparameter settings were then used for the other two bridges (Kruithuisbrug and Lammebrug). Trained models were exported as h5 file type (Figure 5).

4. Model evaluation

4.1 Baseline model

Because machine learning model performance is relative, it is critical to develop a robust and simple model to serve as baseline. The results for the baseline model provide the point from which the skill of all other models trained on the same data can be evaluated. As baseline model, we used the median traffic intensity per minute per weekday. In addition, ARIMA (Autoregressive Integrated Moving Average) and gradient boosting were also tested. Although these models performed relatively well at short-term predictions (1 min. in advance), they did not perform well at long-term predictions (21 min. ahead). Since a good accuracy up to 21 min. is important for the web application, the choice was made to focus on LSTM models. The statistical metrics described in section 2.3 were used to measure performance of LSTM models in relation to baseline models.

4.2 Results

Vector output and encoder-decoder LSTM models were run for each measuring station and each lane. Predicted and observed values were compared for the test set. LSTM models performed better than the median model (Table 2). Accuracy (based on the R^2) was about 20% higher using LSTM models than the median model.

The final model to be used for predicting real-time traffic intensity was then selected for each bridge based on the evaluation metrics. For the Coenecoopbrug and Lammebrug, the encoder-decoder LSTM was chosen for modeling real-time traffic intensity as model metrics were in general better. For the Kruithuisbrug, the encoder-decoder model was only better in one case (station 860 lane 0-1) and, therefore, the vector output model was chosen. An example of the predicted and observed traffic intensity is shown in Figure 2. The univariate time series model learns how to recognize weekend and weekday patterns and makes a prediction based on the preceding traffic intensity. In the web application, three scenarios are presented to the bridge operator: total traffic intensity in the following 1 to 7 minutes, 8 to 14 minutes and 15 to 21 minutes. Accuracy of predictions for the first scenario are presented in Figure 2. One minute ahead predictions are highly accurate ($R^2 > 0.94$) (Figure 3). The accuracy of predictions decreases as the time step increases, although the R^2 is still higher than 0.75. Predictions for the Kruithuisbrug are more accurate than for the other bridges (R^2=0.91 up to 21 minutes ahead). This could be due to the fact that traffic around this bridge is more uniform, leading to fewer outliers.

Table 2. Statistical metrics for the various LSTM models.

Bridge	Station	Lane	Model	MSE	MAE	MAPE	R2
Lammebrug	7	1	Median model	39,92	4,36	39,79	0,66
Lammebrug	7	1	Vector output	20,93	3,31	31,08	0,83
Lammebrug	7	1	Encode-decode	20,37	3,28	32,95	0,83
Lammebrug	8	3	Median model	32,41	4,03	55,62	0,46
Lammebrug	8	3	Vector output	21,09	3,34	49,05	0,65
Lammebrug	8	3	Encode-decode	20,45	3,26	48,16	0,66
Lammebrug	5	5-6	Median model	10,13	2,72	59,21	0,39
Lammebrug	5	5-6	Vector output	7,00	1,93	50,76	0,59
Lammebrug	5	5-6	Encode-decode	6,91	1,91	49,85	0,59
Lammebrug	6	3-4	Median model	36,07	3,90	55,19	0,41
Lammebrug	6	3-4	Vector output	26,55	3,38	53,21	0,58
Lammebrug	6	3-4	Encode-decode	25,64	3,29	49,76	0,59
Lammebrug	9	3-4	Median model	38,13	4,43	56,85	0,24
Lammebrug	9	3-4	Vector output	25,25	3,57	46,58	0,62
Lammebrug	9	3-4	Encode-decode	24,34	3,50	49,80	0,63
Kruithuisbrug	860	0-1	Median model	21,16	3,06	29,30	0,74
Kruithuisbrug	860	0-1	Vector output	18,57	2,98	30,77	0,77
Kruithuisbrug	860	0-1	Encode-decode	18,54	2,97	30,20	0,79
Kruithuisbrug	860	2	Median model	4,50	1,31	36,50	0,61
Kruithuisbrug	860	2	Vector output	4,17	1,35	43,00	0,64
Kruithuisbrug	860	2	Encode-decode	4,20	1,32	38,81	0,63
Kruithuisbrug	861	0-1	Median model	39,29	4,13	30,60	0,71
Kruithuisbrug	861	0-1	Vector output	33,74	3,95	31,60	0,75
Kruithuisbrug	861	0-1	Encode-decode	34,02	4,00	32,66	0,75
Coenecoopbrug	31	0	Median model	14,27	2,35	59,93	0,32
Coenecoopbrug	31	0	Vector output	6,45	1,69	52,24	0,69
Coenecoopbrug	31	0	Encode-decode	6,30	1,66	50,06	0,70
Coenecoopbrug	31	1	Median model	16,57	2,50	61,37	0,27
Coenecoopbrug	31	1	Vector output	7,53	1,78	53,42	0,66
Coenecoopbrug	31	1	Encode-decode	7,47	1,78	53,10	0,67

5. Deployment

5.1 Preparation for production

The prediction model needs to be operational in order to be used in production. The model input variables are obtained real-time from NDW. The model outcome is then available to the web application via an Application Programming Interface (API). Data collected from the API is shown in the web application (Figure 4).

5.2 Real-time data sources

Model input data is obtained from road sensors and converted to traffic intensity per minute (Figure 5). Data needs to be available real-time. For that, the Open Data dataset is col-

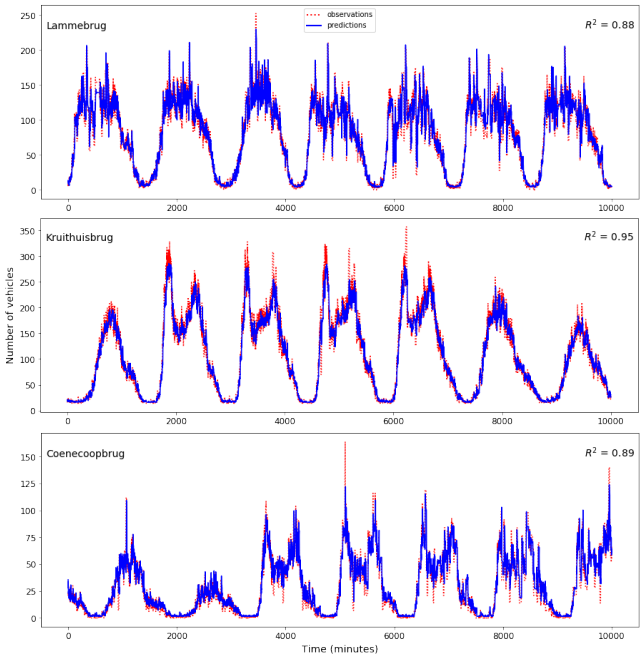

Figure 2. Observed and predicted traffic intensity, measured as number of vehicles in the first 7-minutes scenario. Data: Lammebrug, station 9 (lane 3-4); Kruithuisbrug, station 861 (lane 0-1); and Coenecoopbrug, station 31 (lane 1).

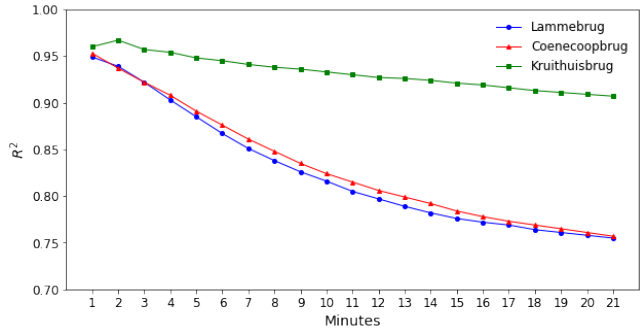

Figure 3. Accuracy of traffic intensity predictions.

lected every minute from NDW via FTP and saved in a Service Bus (Azure EventHub). The Service Bus collects the data per sensor in sets of 30 minutes in a temporary Azure Table for increased performance.

5.3 Prediction API

Model predictions are made available through an in-house API developed in Python [4], based on the FastAPI framework [17]. The choice for an in-house developed API was due to performance issues and to the fact that no standard options are available to develop an API for a Keras model. API documentation was done according to OpenAPI standards.

The following steps, performed by the API, are the result of a /predict/{isrs} call:

1. Get bridge configuration, including the previously loaded bridge-specific LSTM model (.h5 file) and the traffic congestion model (in-memory). The traffic congestion model is developed by Arcadis and determines the vehicle loss hours (VVU) and the recovery time due to a bridge opening based on the shockwave theory.

2. Get the real-time traffic intensity from NDW, including the vehicle type.

3. Run the LSTM model using the real-time NWD dataset as input.

4. Model output, i.e. the predicted intensities during the following 21 minutes, are imported into the traffic congestion model.

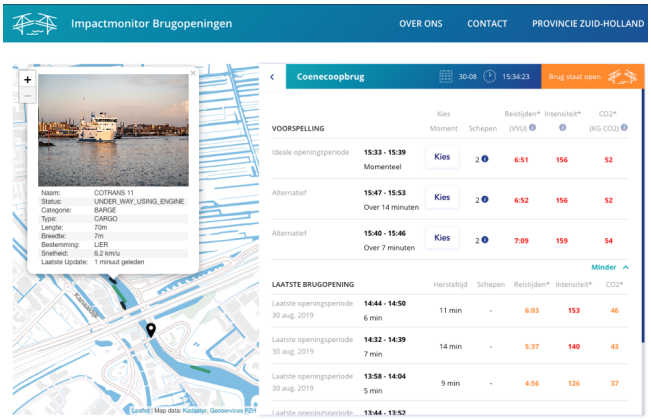

Figure 4. Screenshot of the detail page of one of the bridges monitored in this project.

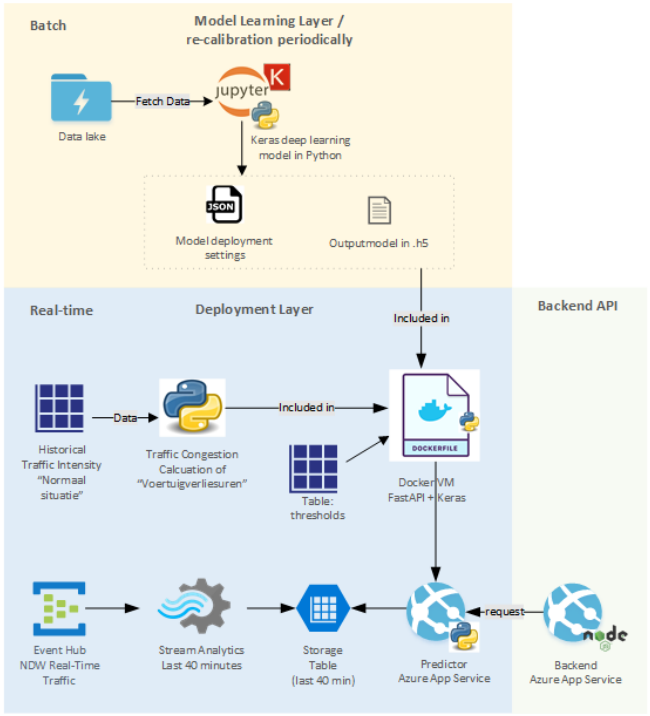

Figure 5. Architecture diagram. The model result (as .h5 file) is used in the real-time prediction API.

5. The traffic congestion model calculates the VVU and the recovery time.

6. CO_2 emission is determined based on the VVU and the vehicle type.

7. Results are presented in three future time blocks of 7 minutes each. These are the time blocks bridge operators can choose to open the bridge.

The prediction API is available via Azure App Services, whereby the logical structure, the traffic congestion model and the FastAPI are packaged in a Docker container including all dependencies. The prediction model is approachable via a web address (https://domain.nl/predict/{isrs}) and returns three time-blocks as result (Figure 6).


```
{
  "predictions": [
    {
      "forecastText": "Ideale openingsperiode",
      "row": 1,
      "delay": 0,
      "offset_start": "13:13",
      "offset_end": "13:19",
      "forecast_voertuigverliesUren": 3.450309388851006,
      "forecast_voertuigverliesUren_color": "green",
      "forecast_TotalRecords_norm": 94.21042148272196,
      "forecast_TotalRecords_norm_color": "orange",
      "forecast_CO2Kilogram_VVU": 26.153345167490624,
      "forecast_CO2Kilogram_VVU_color": "green",
      "herstelTijd": 7.857142857142857,
      "sort": 3.450309388851006
    },
    {
      "forecastText": "Alternatief",
      "row": 2,
      "delay": 7,
      "offset_start": "13:20",
      "offset_end": "13:26",
      "forecast_voertuigverliesUren": 3.5436123423116266,
      "forecast_voertuigverliesUren_color": "green",
      "forecast_TotalRecords_norm": 96.28804349899292,
      "forecast_TotalRecords_norm_color": "orange",
      "forecast_CO2Kilogram_VVU": 26.86058155472213,
      "forecast_CO2Kilogram_VVU_color": "green",
      "herstelTijd": 8,
      "sort": 3.5436123423116266
    },
    {
      "forecastText": "Alternatief",
      "row": 3,
      "delay": 14,
      "offset_start": "13:27",
      "offset_end": "13:33",
      "forecast_voertuigverliesUren": 3.644427254517559,
      "forecast_voertuigverliesUren_color": "green",
      "forecast_TotalRecords_norm": 98.47048182920489,
      "forecast_TotalRecords_norm_color": "orange",
      "forecast_CO2Kilogram_VVU": 27.624758589243097,
      "forecast_CO2Kilogram_VVU_color": "green",
      "herstelTijd": 7.857142857142857,
      "sort": 3.644427254517559
    }
  ],
  "trafficFlow": [
    {
      "roadName": "N207",
      "roadLaneDirection": "W-O",
      "threshold_color": "green",
      "meetlus_id": "PZH01_MST_0031_01",
      "meetlus_index": [1,2,3,4],
      "input": [6, 8, 10, 10, 8, 5, 12, 3, 3, 2, 9, 8, 6, 6, 5, 5, 8, 6, 5,
        5, 7],
      "output": [ 5.9, 6.1, 6.2, 6.2, 6.3, 6.3 ,6.3 ,6.3 ,6.4 ,6.4 ,6.4 ,6.4
        ,6.4 ,6.4 ,6.4 ,6.4 ,6.4 ,6.4 ,6.4 ,6.4, 6.3]
    }
  ]
}
```


Figure 6. Example of the prediction API results for the Coenecoopbrug. The request was done on Saturday 31st August 2019, at 1:13 PM. Three prediction blocks (under predictions), the real-time traffic intensity used as input and the predicted intensity used as output (under trafficFlow) are shown.

5.4 Performance

Application performance is a crucial aspect. Bridge operators do not want to wait long for model results because they monitor multiple bridges and traffic situations at the same time. The Non-Functional Requirement (NFR) is that predictions need to be visible within 3 seconds. The application developed here has a response time between 0.5 and 2 seconds.

6. Conclusion and further research

Traffic intensity analysis showed that several aspects influence traffic around the studied bridges. There is a daily cycle of traffic intensity with a rush hour component. In addition, traffic congestion at a certain moment in time influences the intensity thereafter. Finally, there is a stochastic component due to short strong changes in traffic intensity. In relation to the initially tested median model, which included hour and weekday information, the LSTM model (which has no information about time) results in considerably better predictions. This suggests that not only the preceding traffic congestion but also static factors play an important role in traffic intensity.

Moreover, the use of LSTM models allows predictions over several time steps. In this way, bridge operators have access to predictions of traffic intensity, vehicle loss hours, CO2 emissions and traffic recovery time, due to a bridge opening, up to 21 minutes in advance. There are still possibilities for further optimization of traffic intensity predictions by using a multivariate model, taking into account, e.g., time and weather conditions, or using a larger historical dataset for model training. Additionally, a longer input sequence of traffic can be used and has shown promising results in early experiments. Furthermore, the suggested optimal time window to open a bridge could be optimized by including shipping data.

The model developed in this project is available as a user-friendly web application using big data analysis and recent machine learning developments. This application is ready to be tested by end users.

Acknowledgements

This project would not have been possible without the technical knowledge and support of the department of Infrastructure Management Services (DBI), especially Ellen van der Knaap, Tim Blanken, Tino van As and Jillis Mani. We are thankful to many colleagues from the vDWH team of the Province of Zuid-Holland for their contribution to the development, deployment and streamline of this application, especially to Bart Witteveen, Firuze Kuru, Dennis Weijsenfeld en Kevin Otjes.

References

[1] https://bereiknu.nl/ BEREIK! is hét samenwerkingsplatform van Rijkswaterstaat, Provincie Zuid Holland, Metropoolregio Rotterdam Den Haag, Gemeenten Rotterdam en Den Haag en Havenbedrijf Rotterdam voor regionale bereikbaarheidsvraagstukken.

[2] https://blauwegolfverbindend.nl/, Blauwe Golf Verbindend ontsluit real-time gegevens over geopende bruggen en beschikbare ligplaatsen in havens

[3] https://www.zuid-holland.nl/@21238/smart-shipping/ Partnershipovereenkomst Smart Shipping.

[4] Van Rossum, G., Drake Jr, F.L., 1995. Python tutorial. Centrum voor Wiskunde en

Informatica, Amsterdam, The Netherlands. https://www.python.org/downloads/release/python-367

[5] Abadi, Martín, et al. "Tensorflow: A system for large-scale machine learning." 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). 2016.

[6] Chollet F. , 2015. Keras, GitHub. https://github.com/fchollet/kera

[7] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." Journal of machine learning research 12.Oct (2011): 2825-2830.

[8] Zurada J.M.,1992. Introduction to artificial neural systems vol. 8. West publishing company, St.Paul.

[9] Jain A.K., Mao J., Mohiuddin K.M., 1996. Artificial neural networks: A tutorial. Computer 29, 31-44.

[10] Zhang G., Patuwo B.E., Hu M.Y., 1998. Forecasting with artificial neural networks: The state of the art. International journal of forecasting 14, 35-62.

[11] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521.7553 (2015): 436.

[12] Mandic D.P., Chambers J., 2001. Recurrent neural networks for prediction: learning algorithms,rchitectures and stability. John Wiley & Sons, Inc.

[13] Hochreiter S., Schmidhuber J., 1997. Long short-term memory. Neural computation 9, 1735–1780.

[14] Gers F.A., Eck D., Schmidhuber J., 2001. Applying LSTM to time series predictable through timewindow approaches. In: Dorffner G., Bischof H., Hornik K. (eds) Artificial Neural Networks. ICANN 2001, Lecture Notes in Computer Science, vol 2130. Springer, Berlin, Heidelberg.

[15] Sak H., Senior A., Beaufays F., 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. INTERSPEECH 2014, 14-18 September, Singapore, 338-342.

[16] Brownlee, Jason. "Text Generation With LSTM Recurrent Neural Networks in Python with Keras." (2018).

[17] https://fastapi.tiangolo.com/ FastAPI framework, high performance, easy to learn, fast to code, ready for production

[18] Newell, Gordon F. "A simplified theory of kinematic waves in highway traffic, part I: General theory." Transportation Research Part B: Methodological 27.4 (1993): 281-287.