Data Science Submission

Edition: 2023 2A

Project: Transport

Primary Topic: DM

Secondary Topic: DEP

Members: Miguel de la Cruz Cabello, Bas Vreeman

Last Update: 25/04/2024, 23:15:31

1. Motivation

Transportation is about moving people and goods from A to B. Being able to transport people and goods is a prerequisite for economic growth. The increasing complexity and unpredictability of urban traffic systems present significant challenges to transportation management and public safety. Rapid and accurate incident detection is crucial for implementing timely interventions, reducing congestion, and ensuring efficient traffic flow. Traditional incident detection methods often rely on manual observation or predefined rules, which may not be sufficiently adaptive or scalable to handle the dynamic nature of traffic systems (Wang et al., 2016).

Our business question, "To what extent can a convolutional neural network, trained on historical traffic flow data, determine if an incident on the Ring Rotterdam has occurred?" addresses a critical gap in the current incident detection methodologies. Leveraging the power of machine learning, particularly convolutional neural networks (CNNs), offers promising avenues to enhance incident detection accuracy and efficiency. CNNs are well-suited for analyzing spatial and temporal data patterns, making them ideal candidates for this task.

Our approach to developing spatiotemporal diagrams as inputs to the neural network is innovative and holds substantial promise. These diagrams encapsulate the essential features of traffic flow, incident locations, and their temporal relationship, providing rich and contextual information to the CNN. Training the CNN on such comprehensive and structured data can enable it to recognize intricate patterns and anomalies associated with incidents more effectively.

2. Business Question

Our project seeks to answer the following business question using the SMART criteria: " To what extent can a convolutional neural network, when trained and built and delivered with generated historical traffic flow data within a period of 10 weeks, determine if an incident on the Ring Rotterdam has occurred, with a high accuracy on new data?". We believe this question could address the problem of determining traffic incidents on a specific road segment using spatiotemporal diagrams representing loop detector positions and velocities over time.

The approach using the SMART criteria was the following:

- Specific: As mentioned above, our question targets the specific challenge of determining if an incident has occurred on Ring Rotterdam, using spatiotemporal diagrams of speed flows.
- Measurable: The success of our project is quantifiable through the accuracy of the CNN determining if an incident occurred or not based on the spatiotemporal diagrams. We measure performance through validation accuracy / newly seen data.
- Achievable: the work behind the business question consists of existing data from loop detectors and incidents in Ring Rotterdam, processed into 74 incident and 74 no-incident spatiotemporal diagrams. We use a CNN which aligns with the achievable goal of enhancing traffic incident detection through machine learning.
- Relevant: We believe that this business question is highly relevant to traffic management authorities that need reliable methods to determine traffic incidents as fast as possible to reduce congestion and improve road safety for example.

- Time-bound: we set a period of 10 weeks.

3. Source Data

Ensuring the quality of data is crucial for any data-driven project. Even with advanced analytics and modeling techniques, poor data can lead to inaccurate and unreliable results. In this section, we will discuss our source data, the data quality issues we encountered, the steps we took to detect and address them, and the impact that these actions had on our project. All data was provided by NDW. It consisted of:

- Speed flow data from loop detectors (1-minute aggregates CSV-files) for two time periods of the year 2020 (March 1st to 4th April and April 27 to May 31). What should be noted is that this was during the COVID-19 pandemic.
- Status information (CSV-files) in occurrence for the whole Netherlands for different time periods for the year 2020. It included opening bridges, road works, incidents, and traffic jams.
- Travel times of predefined routes (CSV-files) from April 27th, 2020, to May 31st, 2020.
- Cycling data providing counts, 1-minute and 1-hour aggregates (CSV-files).

Nevertheless, for the purpose of our business question, we focused primarily on the speed flow and incident dataset. The goal of the data processing is to get data that can be used to generate the spatiotemporal diagrams later. What is needed for this is data of different road sections, for certain timespans. An equal number of diagrams needs to be created for incident cases as for no-incident cases. More on this is explained in the Method Section.

To realize this, the initial step was to load the speed flow metadata, focusing on the detector IDs, location coordinates, street name, and lane of the road. We filtered the area of the city to Ring Rotterdam, due to its strategic location of the road network and the high traffic volume throughout the day. We then loaded them into another data frame the incidents dataset, focusing on the incident IDs, start time, location coordinates, type, and detailed type of the incident. Filtering the incidents from March 1st until May 31st was more challenging since we did not have the street names in the dataset. Instead, we drew a polygon shape with the max location coordinates of Ring Rotterdam to determine the edges of the rectangle. With this method, we were able to exclude incidents outside the polygon. We then defined each size of the rectangle as a different segment and made a 'Segment' column on both datasets to determine which incidents and detectors belonged to which segment (by segmenting the road network into manageable segments, we could better understand the spatial context of incidents to nearby detectors when mapping). Once we had filtered incidents and detectors to Ring Rotterdam, we proceeded to the following stage. For better mapping and efficiency of the algorithm, we decided to primarily focus on incidents and detectors on a single lane in a clockwise direction. To accomplish this, we developed an interactive map where we could add detectors and incidents to separate lists for later exclusion by clicking on them. At this stage, the data had to be cleaned manually as we had no reference in the data frames. We ended from 2,100 incidents and 644 detectors in Ring Rotterdam to 266 incidents and 109 cameras in a single lane and a clockwise direction for Ring Rotterdam. *Note: Continue reading the data exploration section in Appendix A (due to the lack of words available in this box)*.

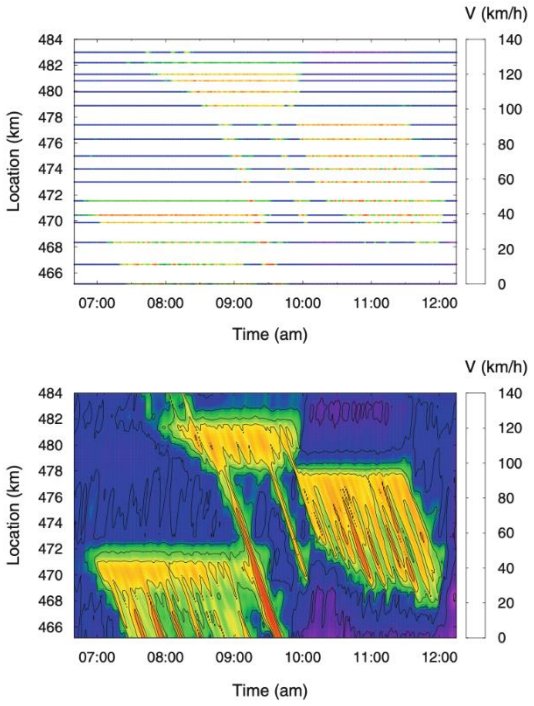
4. Method

Method

The dataset contains speed flow data, locations of cameras, and locations of incidents. What we wanted to analyze initially was incident data, to potentially predict or detect whether incidents are going to happen or happened. However, we perceived this also more as a time series analysis problem, which is not something we learned to do from any courses yet. To solve this, we made use of space-time diagrams, which show a traffic situation over multiple hours at a road segment at one glance, which eliminates the need for a time series analysis.

The space-time diagrams are image data. Thus, we need to use a convolutional neural network to train the model. To train a convolutional neural network, quite some data is needed, therefore we generated a labeled dataset of 74 space-time diagrams of incident traffic situations, and 74 diagrams of no-incident traffic situations, which might be considered the bare minimum.

The diagrams have been created using spatiotemporal interpolation, which is a method described in Treiber (2013). The goal is to create from the limited amount of data shown in the upper picture in Figure X, the bottom picture, by estimating the velocities at the white places, where loop detectors were absent.



To implement the spatiotemporal interpolation, we used the formulas from Treiber and implemented them in our program. The formulas with their definitions of variables are given in the figure below, which is copied from the book:

The basis of spatiotemporal interpolation and smoothing is a *discrete convolution* with a kernel ϕ_0 that includes all data points i:

$$V(x,t) = \frac{1}{\mathcal{N}(x,t)} \sum_i \phi_0 (x - x_i, t - t_i)\, v_i. \tag{5.1}$$

In principle, we can use any function as the weighting kernel $\phi_0(x,t)$. To avoid artifacts, the kernel should have following properties:

- It should be localized, i.e., $\phi_0(x,t)$ tends to zero for sufficiently large values of $|x|$ and $|t|$.
- The maximum of $\phi_0(x,t)$ should be at $x = 0$ and $t = 0$.
- $\phi_0(x,t)$ should be a continuous function of x and t.
- $\phi_0(x,t)$ should be monotonically decreasing with $|x|$ and $|t|$.

For our purposes, the symmetric exponential has proved itself useful[1]:

$$\phi_0(x - x_i, t - t_i) = \exp\left[-\left(\frac{|x - x_i|}{\sigma} + \frac{|t - t_i|}{\tau}\right)\right]. \tag{5.2}$$

Here, σ and τ are the smoothing widths in the spatial and temporal coordinates, respectively. The denominator $\mathcal{N}$ of Eq. 5.1 denotes the normalization of the weighting function, given by the sum of all discrete weights:

$$\mathcal{N}(x,t) = \sum_i \phi_0(x - x_i, t - t_i). \tag{5.3}$$

The code for this can be found in Appendix B, or in GitHub: https://github.com/migueldlcc/DataScience/blob/main/basFiles/3hoursFromIncidentToInterpolationData%20copy/3hours_spatio_interpolation.ipynb .

The sigma and tau are parameters that can be tuned to liking. We have trained the model with two sets of generated images, one where the sigma was 200, and one where the model was 500. At first, the sigma of 200 was determined based on a visual examination of the graphs. However, due to the performance of the CNN in those diagrams, another model training has been done with images created with Sigma 500. This made no difference. The tau is set to 45.

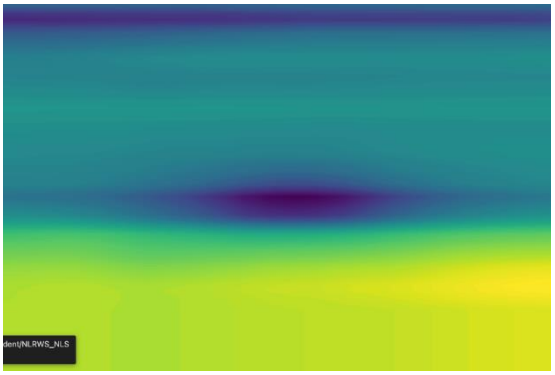
In total, 74 graphs were created for incidents, and 74 graphs were created for no incidents. See the appendix for examples of the graphs. More can be found at https://github.com/migueldlcc/DataScience/ .

CNN

The exploration and preparation of the data and building the algorithm for the generation of the diagrams were time-consuming. Due to this only 1.5 days were left for building the CNN. Thus, a simple model was built exploiting ChatGPT mainly, which ran 12 epochs over the data, with a batch size of 32, without using cross-validation (due to time constraints), with a learning rate of 0.001, see Appendix C for the architecture of the model. Other hyperparameter settings have been tested.

Since the axis are the same for all diagrams, they have been removed. Thus, the input only contains the colors which represent the speeds. See the figure below for an example. Later, the images were gray-scaled, as this did give the best-performing model (this is not reflected in the image below).

To evaluate the CNN, several metrics have been used, namely the accuracy, precision, specificity, and the f-1 score.



5. Results

Results from CNN with sigma = 200, lr = 0.001

Because we wanted the model to focus especially on speed changes, without putting the focus on true speed, we tested using grayscale images as input, as well as normally colored inputs. For grayscale:

Some stats for the model:

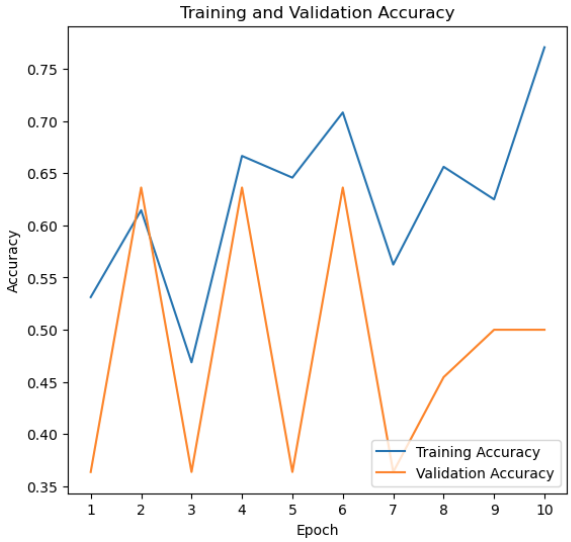
- The model was able to get a validation accuracy of 64%.
- However, this was more luck than wisdom, as you can see from the graph that it was only predicting accidents. See the confusion matrix below.

Metric	Value
True Positive	14
False Negative	0
False Positive	8
True Negative	0

- For the non-grayscale, it performed worse, with a validation accuracy of 50%. See the confusion matrix below.

Metric	Value
True Positive	3
False Negative	11
False Positive	0
True Negative	8

What is visible from the confusion matrix presented above is that the model does not always predict one of both and can modify weights based on the data. Also, the graph below, which shows the training and validation accuracy over time, shows that the model can learn based on the data, as the training accuracy increases to 75%. However, the validation accuracy does not follow, which is a sign of overfitting.



Although the model trained on grayscale input images performs best, it does so by choosing only positive, or only negative, based on the training data. Due to a lack of time for tuning hyperparameters, or altering the architecture of the CNN, we will focus on this result for further analysis of the grayscale model, having an accuracy of 64%.

Performance metrics of the grayscale model

The precision (64%) measures the accuracy of positive prediction, so this is the number of positive predictions, over all predictions. Thus, the precision is the same as the accuracy, which is 64%.

Recall (100%) measures the ability of the model to find relevant cases. So, this is the true positives over the true positives + false negatives. In this case, the model did not predict any negatives, so this measure equals 1.

F1 Score (78%), The F1 score is the harmonic mean of precision and recall, providing a single score that balances both the model's precision and recall. An F1 score of about 0.778 is high, which indicates a good balance between precision and recall. However, in this context, it is driven by the extremely high recall, not the precision.

Looking at those 3 measures, you expect a well-performing model. However, the specificity exposes that the model is malperforming. The specificity of 0 shows that the model did not predict any negative (no-incident) case correctly. This then leads to a specificity of 0, since the specificity is the true negatives, over true negatives + false positives.

6. Reliability of results

The reliability of spatiotemporal diagrams generated

Late in the process, we found that there was a difference between certain distance calculations. For the distance from loop detectors to the incident, the direct distance ('helicopter distance'), from

each incident to the start of the trajectory was considered, however, when creating the random road segments that contained no accidents, the distances were calculated from camera to camera, due to which the shape of the road was followed better. Especially in turns on the road, this might have caused some morphed data.

Some graphs are a bit off due to a lack of camera coverage on certain road sections. By manual analysis, this is estimated to be the case for around 5% of the diagrams. Lastly, what should be noted is that no space-time diagram has been created which looks exactly like the book examples.

To conclude, we generated space-time diagrams for incidents and no-incidents at all times of the day, including non-peaking hours. However, traffic jams and large speed decreases are unlikely to occur at lower traffic density times, unless an extremely heavy incident happens. Due to this, it might be impossible for anyone, including a CNN, to distinguish between traffic conditions.

CNN

Data Limitations: The dataset used was small, and the model ran only 12 epochs, which might not have been sufficient to achieve optimal learning. Also, no cross-validation had been performed. Additionally, the batch size and learning rate were not experimentally optimized, which might have affected the model's performance.

Overfitting and Performance Issues: the CNN tended to overfit, as indicated by training accuracy which did not translate into improved validation accuracy. Additionally, although the model got to a validation accuracy of 64%, this was mere luck since it ‘gambles’ on either incident or no incident for all predictions.

7. Technical depth

The first thing, which goes beyond our primary topic, is the implementation of the convolutional neural network. Prior knowledge was available due to previous courses in Machine Learning I and Machine Learning II. Also, those courses helped with translating the formulas from Traffic Dynamics by Treiber into functional Python code.

On top of that, we had to create an interactive map, to be able to easily select and remove data points from the map of Rotterdam. To do this, we used a combination of HTML and JavaScript, which we both have experience with through personal and prior programming projects.

8. Conclusions & Recommendations

Summary

During this study, a balanced dataset of 74 incident-, and 74 no-incident situations is created. The training of the CNN on the data did not show promising results, as it appeared to either predict all on incidents, or all on no-incidents after training. Also, the model seemed to overfit, considering the validation accuracy which remained low, while training accuracy increased.

Multiple things could have led to the poor performance of the model. Firstly, the generated graphs might not reflect the traffic situations well or might be insensitive to incidents. What might have contributed to this insensitivity is a low traffic density due to COVID-19, which might have led to such a low density, that incidents do not slow down the traffic.

Also, we did not test the spatiotemporal diagrams to a baseline, to verify that they are generated correctly. They have only been observed to be valid by manual examination.

What we recommend to our stakeholders, which are the municipality of Rotterdam, emergency services in Rotterdam, and the NRW (provider of the data), to continue researching the possibility of using CNNs to analyze historical traffic data via space-time diagrams. We could not provide a conclusive answer on its effectiveness, due to a lack of data, or a potential lack of quality thereof. So, it is advised to increase the dataset, preferably with data outside of covid-times, and spend more time on the tuning of hyperparameters and the network architecture.

Also, as a last remark, it was mentioned by Estefania during our presentation, that to reduce the preprocessing, we could potentially also directly use the speed flow data, without interpolating, in a neural network, by transforming the data into vectors. This approach should also be considered in the future.

9. Reflection

We started by planning a conversation with the project owner: Luc Wismans, to ask about possibilities with the dataset, and whether more data was available. Then we started exploring the dataset, while also reading the relevant book chapters in Traffic Flow Dynamics by Treiber, to get an idea of the possibilities with the dataset.

We wanted to do something with predicting or detecting traffic situations, so we were happy to find the incident data.

No big issues occurred, besides the lack of time for the creation and tuning of the CNN during the project. Also, with more time, we would have done a new iteration of generating more datasets, using only peak hours (e.g. 16:00 – 19:00) first, with data that was not recorded during the pandemic.

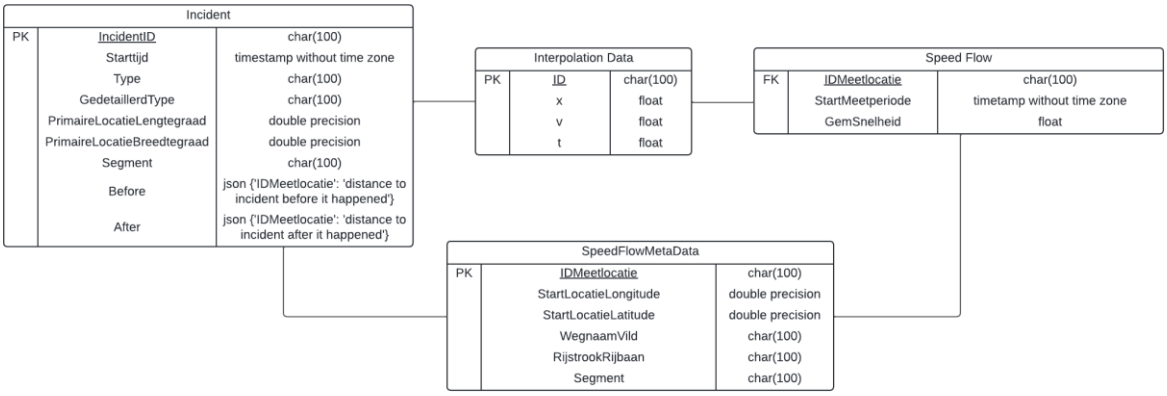
The skills learned during this course were perfectly applicable to our project. We were provided with exactly the right toolbox to get going and broaden our knowledge in data science.

Furthermore, it was useful that we both followed Machine Learning I and II before, due to which implementing the spatiotemporal interpolation formula was within the range of our capabilities. Also, previous coding experience might have helped us with some challenges, such as creating an interactive map to distill the relevant incident and camera locations.

Lastly, GPT has been used in some cases. For example, there was a piece of code, which had to be kind of reverted, and typing this manually would have become a tedious task. To save time, a part of the program was produced up until the part after which the reverted code needed to be added. Then GPT created the rest. Also, GPT was the first thing to consult if a bug could not easily be solved. If GPT could not give a useful answer, Google was used.

10. Cube Data

The schema presented below offers a different approach to constructing a model for incident analysis and traffic data interpolation. This model, which does not conform to a conventional star schema, pivots around a central fact table that has been designed from scratch, capturing interpolation data derived from incidents and speed flow measurements. However, this design facilitated us for efficient data retrieval and later generation of the spatiotemporal diagrams. As explained in the data source section, the incidents were mapped to loop detectors by leveraging the coordinate locations provided within the speed flow metadata. For each incident, the 'Before' and 'After' columns in the incident dataset were populated with dictionaries. These dictionaries recorded corresponding detector mappings, utilizing detector IDs as keys and their respective distances from the incident locations as values. On the other hand, the speed flow metadata was correlated with the speed flow data to obtain average speeds and recording timestamps. This data, covering both sections with incidents and those without, formed the basis of the data interpolation table. The table encapsulates variables such as average speed, timestamps, and distances from the recording's start point for a total of 148 road sections. We use this table to generate the spatiotemporal diagrams that train the convolutional neural network.



11. DM Issues

In the process of developing the spatiotemporal diagrams, particular data management challenges were encountered and addressed to ensure the integrity and utility of these diagrams.

A key issue that we found was the presence of negative values within the average speed data. We supposed that such values were indicative errors of data collection or processing since negative speeds are not feasible in real-world traffic scenarios. To solve this, any record showing a negative average speed was removed from the dataset, preserving the statistical significance for the generation of spatiotemporal diagrams.

Another challenge was the limited amount of data (spatiotemporal diagrams) to train the model. A small dataset can severely hinder the model's ability to learn and generalize, especially in complex situations such as the problem we dealt with in our project. The reason we had a limited amount of diagrams (148) was because we spent most of the project cleaning and preparing the data. The filtering of incidents and loop detectors took a lot of time as we had to create an interactive map and remove them manually.

12. References

Github repository: https://github.com/migueldlcc/DataScience/tree/main

Treiber, Traffic Flow Dynamics, 2013.

Wang, Y., Zhang, H., & Liu, Q. (2016). A Survey of Traffic Incident Detection Methods. IEEE Transactions on Intelligent Transportation Systems, 17(9), 2637-2654. Retrieved from https://dblp.org/db/journals/tits/tits17.html

13. Appendix

Appendix A: Data Exploration Continued

These steps were crucial to understanding the quality and scope of the data before proceeding to the mapping process.

To map incidents to detectors we used a proximity analysis where we calculated the distances between the detectors and the incidents to identify which detectors are within a certain range of an incident (2 km before and 5 after an incident). We first converted the coordinate locations to radians and used the Haversine formula to compute the distances between the incident and detector locations, considering the curvature of the Earth:

$$a = \sin^2\left(\frac{\Delta lat}{2}\right) + \cos(lat_1) * \cos(lat_2) * \sin^2\left(\frac{\Delta lon}{2}\right)$$

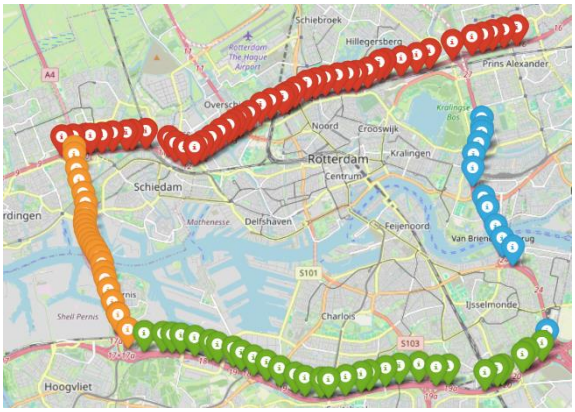
$$c = 2 * atan2\left(\sqrt{a}, \sqrt{1-a}\right)$$

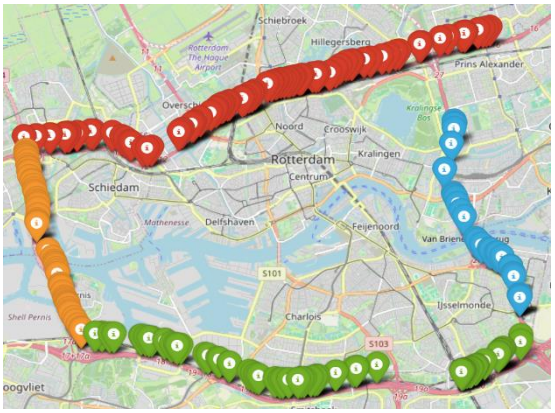
$$d = R * c$$

Where:

- $\Delta lat = lat_2 - lat_1$
- $\Delta lon = lon_2 - lon_1$
- lat_2 *and* lat_1 are the latitudes of the two points in radians.
- lon_2 *and* lon_1 are the longitudes of the two points in radians.
- R is the radius of the Earth (mean radius = 6,371 km).
- d is the distance between the two points

We created 'Before' and 'After' columns to the incidents data frame containing dictionaries of detectors that belonged to each incident (the key was the detector ID and the value was the distance to the incident in kilometers). To check for correctness the mapped detectors were compared with a selected subset of incidents to check if they were indeed the ones closest to the incident location. We visualized the subset of incidents in blue and the mapped detectors in red and green for 'Before' and 'After' the incident. Later, this data frame was added to the database for further data formatting. The first figure shows the mapping of an incident (blue) to its detectors (red representing before, and green representing after), the second figure shows loop detectors in Ring Rotterdam, and the third figure shows incidents in Ring Rotterdam.





Now that each incident was connected to a detector ID the goal was to capture speed data on the corresponding road section where the incident occurred. By retrieving the detector's ID from the 'Before' and 'After' columns we were able to get that data from the speed flow dataset which included the average speed 'v', and time 't' of the observation (1 hour before the incident happened until hour 3). We took road sections of 6,000 m in total, 2000 m before the incident happened and 4,000 after it happened. In addition, we included a distance 'x' of each detector from the starting point of the road section where the incident occurred (an incident was always set to 3,000 m). This distance 'x' of each detector to the starting point can be calculated as follows:

$$x = 3000 \pm \text{distance to incident}$$

By formatting the data in this manner, we could effectively analyze the relationship between incidents and the corresponding speed data captured by the detectors on the road sections. This structured approach ensures that we have detailed and relevant data points for each incident, facilitating the construction of spatiotemporal diagrams trained on a CNN to gain a more in-depth and accurate understanding of whether or not an incident happened, but will be explained later in the Method section.

Throughout the exploration process, several data quality issues were identified and addressed to maintain the integrity and reliability of the mapping:

- Longitudes and latitudes were converted to float datatypes for better manipulation.
- Start times were converted to date datatypes for filtering.
- Duplicate entries on ID and coordinate location, and also simply on coordinate locations were removed.

- Missing values or empty cells were dropped too.

Data exploration code can be found here:
https://github.com/migueldlcc/DataScience/tree/main/miguelFiles

Appendix B: Code for spatiotemporal diagrams

```python
def phi_0(x, t, xi, ti, sigma, tau):
    return np.exp(- (np.abs(x - xi) / sigma + np.abs(t - ti) / tau))

def N(x, t, datapoints, sigma, tau):
    return sum(phi_0(x, t, xi, ti, sigma, tau) for xi, ti, _ in datapoints)

def V(x, t, datapoints, sigma, tau):
    normalization_factor = N(x, t, datapoints, sigma, tau)
    return sum(phi_0(x, t, xi, ti, sigma, tau) * vi for xi, ti, vi in datapoints) / normalization_factor

# takes in datapoints, which is a list of tuples of the form (x, t, v) for each datapoint
def spatiotemporal(datapoints):


    # Create a grid for the spatiotemporal map
    spatio_map = np.zeros((int(minutes / minute_steps), int(meters / meter_steps)))

    for i in range(0, minutes, minute_steps):
        for j in range(0, meters, meter_steps):
            spatio_map[int(i / minute_steps), int(j / meter_steps)] = V(j, i, datapoints, sigma, tau)

    return spatio_map
```

Appendix C: CNN Architecture + hyperparameters

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	320
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	0
flatten (Flatten)	(None, 115200)	0
dense (Dense)	(None, 128)	14,745,728
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129

Total params: 44,515,589 (169.81 MB)

Trainable params: 14,838,529 (56.60 MB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 29,677,060 (113.21 MB)