Data science report

Predicting road safety using map data
Primary Topic: DM, Secondary Topic: DPV
Course: 2020-1B – Group: 53 – Submission Date: 2021-07-13

J.G. Hamoen
University of Twente
J.G.Hamoen@student.utwente.nl

K.J. Rijnbergen
University of Twente
K.J.Rijnbergen@student.utwente.nl

ABSTRACT

Safe roads and road networks are of great importance in our society, as they help us safely and quickly get to where we need to go. Designing as well as evaluating these designs, however, is not a trivial task. Prior research has explored the classification of traffic accidents, but analysing the danger levels of road networks by means of image classification has not been done yet, at least not to our knowledge. In this paper, we present our take on a machine-learning model that can classify maps of our road network based on the number of accidents that occur at a certain location. This way, we can provide road designers and city planners with a way to evaluate their design before the implementation has even started.

KEYWORDS

Image classification, Road Networks, Traffic, Junction, Road, Danger, Safety, Support Vector Machine, Data Mining, Data Preparation

1 INTRODUCTION

Transportation planning can be difficult at times. Designs for junctions, highway exits, and other situations may look safe during design, but once implemented, it may turn out to be fairly dangerous. In this report, we attempt to create a model that can predict the danger level of a road network (more specifically, part of a road network), based on an image of what the network will look like once implemented. This way, we can evaluate whether or not the design is ready to be implemented, or if we need to get back to the drawing board.

2 BACKGROUND AND RELATED WORK

In order to design a tool without having to re-invent the wheel, we start out by taking a look at the current situation and related works. This information, as found in the following paragraphs, serves as the starting point of our research.

2.1 Background

When looking at simple road networks, humans are able to estimate the danger level of situations fairly well. We identify blind corners as situations where we have to be careful, while single-lane high speed roads are often fairly safe, despite the high speed. For more complex road networks, this estimation is more difficult. For Multi-lane intersections, this estimation can vary depending on several factors. Inadequate traffic signs, blind spots, or merging lanes are all factors that can cause confusion in drivers, ultimately leading to more frequent accidents and reduced overall safety levels.

The road design is responsible for a large part of present hazards, and thus, the accidents that occur. These accidents only happen once the road is built, meaning we can only start collecting relevant data on safety once the road is already built, and any changes that need to be made will have a high cost attached to them.

The main goal of this work is to propose a method we can use to evaluate the safety level of a road before we build it, not afterwards. This way, we have an indication beforehand whether a road design is dangerous or not, meaning we can prevent construction of unsafe roads, increasing overall safety and lowering potential future costs.

2.2 Related Work

Closest related to our research is that of Princess et al. [2], where images of road accidents are classified based on their severity and type of accident. Features of the images are extracted and provided to both the support vector machine (SVM), and the k-nearest-neighbor (KNN) algorithm.

In our research, we take a similar yet different approach. The idea of image classification for analysing road safety and accidents seems promising. Unlike in Princess et al., however, we use images of the road network designs to analyse overall safety, instead of analysing individual accidents that occur.

3 DATA SET

In our study, we used a Data Set that covers traffic accidents related to the export of goods via the city of Rotterdam, the Netherlands. The Data Set contains data from March 1st, 2019 to August 31th, 2019. This timeframe is outside the Covid-19 pandemic, as such the data should reflect typical use of the road network. The pandemic should hopefully recede soon, and as such this work is aimed at a normalised traffic situation. For a detailed overview of all data columns, see Table 3 in the Appendix. The raw Data Set contains 161962 rows of data, aggregated from different sources like the ANWB (Dutch version of the US AAA) and different Dutch (local) authorities, with also multiple versions of each accident. There are some columns with secondary locations which have rows with no data point, but those are not used in our project. [5]

As can be seen in 1, the locations in this Data Set are concentrated around Rotterdam. This is expected, because the Data Set this table originates from is itself focused on Rotterdam.

4 METHODOLOGY

This section describes the methodology of preparing the accident data, selecting the required features, and mining map data for use in two classifiers.

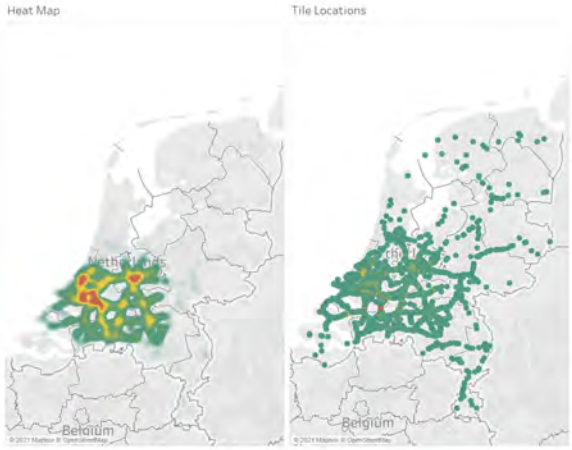


Figure 1: A density map of the locations listed in the Data Set (left), and all points plotted on the map (right).

4.1 Feature Extraction & Labelling

We identified the coordinates of traffic accidents on the map. Our map has been divided in tiles, some of which have more accidents than others. The tiles are labelled based on their number of accidents , possible labels being : "very_low", "low", "medium", and "high". We extract images of these tiles using the OpenStreetMap API [6] in Python. Some examples are provided below.

We separated the Data Set into four classes, being "very_low" [0, 1] accidents, "low" [2, 3] accidents, "medium" [4, 10] accidents, and "high" [11,) accidents.

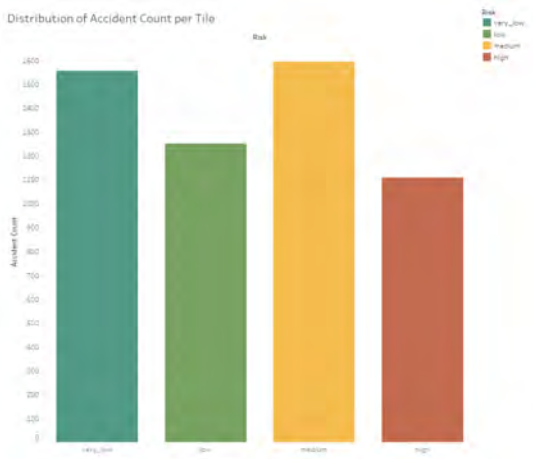
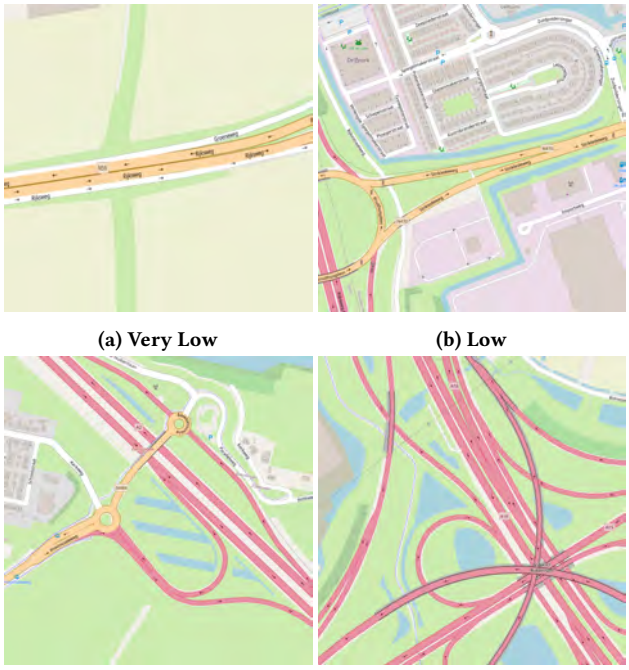


Figure 2: Distribution of classes in our Data Set.

4.2 Aggregation

Initial inspection of the Data Set with our initial categorised data revealed a mismatch between the data and reality. As can be seen in Figure 4, the granularity of the geospatial data is quite high, which can be a good thing. However, this clearly dangerous intersection



(a) Very Low **(b) Low**

(c) Medium **(d) High**

Figure 3: Examples of roads and their respective danger levels.

(knooppunt ridderkerk) would be classified as having a very low risk many times. As such, aggregation of the data was deemed necessary. K-Means clustering was tries, but was not usable for our application, because the clustering would result in a limited number of data points. For more on k-means clustering, see the discussion.

The final aggregation method uses the map tiles native to OpenStreetMap. All data points are clustered to each tile, and the tile is then used in the classifier. This aggregation method has good synergy with our data mining method, as the OpenStreetMap API serves tiles anyhow. Using this aggregation method leads to more appropriate results, while maintaining a sufficient granularity. The same junction with aggregated data can be seen in Figure 5. As for the database-part of this process, we have used the intermediate variables: "tile_x" and "tile_y" in order to join our Location_ids to the tile_ids.

4.3 Map Data Mining

With the aggregated data on the scale of the OpenStreetMap map tiles, the image Data Set can be constructed. For each location in the locations table, represented as can be seen under "Locations" in Figure 6, we use smopy [8] to mine GPX data from the openStreetMap API and save the map tiles as PNG images. Then the images are split into test, train and validation sets, and moved to corresponding folders according to their risk category, to be used in a classifier or regression model.

Figure 4: The unaggregated datapoints on a notoriously dangerous Junction in the Netherlands, all the blue points have a "very_low" risk label.

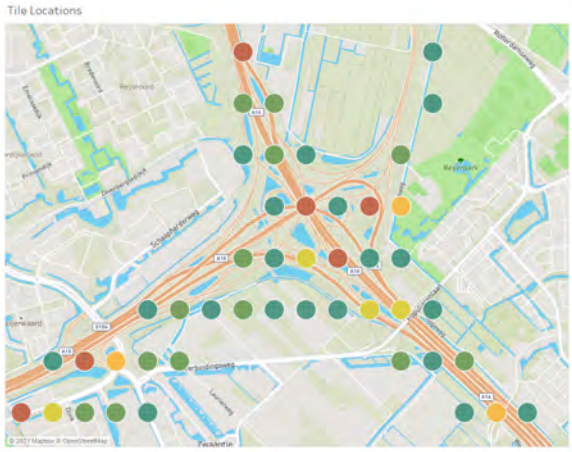


Figure 5: The same Data as Figure 4 after aggregation

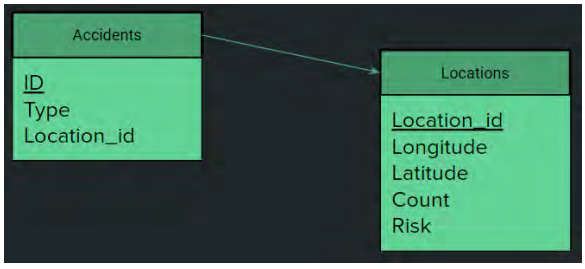


Figure 6: Star schema for our Data Set.

4.4 Classifiers and Evaluation

To apply and generalise the knowledge in the acquired Data Set, two machine learning methods are used: A support vector regression (SVR) model and a Convolutional Neural Network.

4.4.1 Support Vector Regression. The Support Vector Regression algorithm (SVR) is a modified version of the support vector machine (SVM)[4], a machine learning classifier that separates classes by fitting a hyperplane that maximises the margin between the plane and its closest points (i.e. the support vectors). The SVR is different from the SVM in the sense that it does not predict a class which the image belongs to, but a number that corresponds to the situation present in the image. (In our case, the number of accidents occuring at a certain location in a certain timeframe). In our studies, we use the scikit-learn python library for our SVR [7], using a linear Kernel, and C=100. The advantage of a SVR is that it is relatively efficient and easy to train. It takes less computing power than a neural network, and usually reaches significant results relatively quickly. The disadvantage of a SVR over a neural network is that the peak performance given more compute is lower. When given large amounts of computing power, neural networks usually outperform SVR systems.

The advantage of using regression over classification is that it usually performs better on continuous variables, and that even when the classification is off, the regression system can still be in the same ball park figure range. And for this use case that could be preferable. The difference between 50 and 60 predicted accidents would be arbitrary, as the urban planner using this system would need to take a good look at his new intersection anyhow. The model is evaluated using the R2 scores, mean absolute error, and mean squared error. These are common evaluation metrics used for regression models, and more information can be found on the sklearn wiki [7].

4.4.2 Convolutional Neural Network. The other model we created is based on the convolutional neural network (CNN). CNNs are a class of deep neural networks, often used in image classification [9]. CNNs are regularised versions of the multilayer perceptron, a very basic type of neural network. An illustration of the CNN is provided in Figure 11 in the Appendix. This particular network contains several convolution layers, which perform linear operations that involve multiplying a set of weights with our input (which consists of the color values of the pixels in our images. In addition to convolution layers, Max-pooling layers (which downscale images by taking the maximum value of a subset of pixels, as illustrated in Figure 7), and output layers (which do the actual classification) are also present. Unlike the SVR, which predicts a number of accidents to occur, our CNN identifies the danger level, or, "class" of our input images.

We opt for a convolutional neural network as they are well suited for the analysis and classification of images. They can often attain very high accuracies, which is exactly what we are after. As with most things in life though, there is no such thing as a free lunch. Convolutional neural networks are extremely computationally intensive, as they often use complex mathematical operations on very large scales. A lot of parameters need to be trained, (even a fairly simple model like ours requires about 30 million parameters to be trained (29,985,988 to be exact)), and as you can imagine, the model needs a lot of training data to do so.

The model we use is a sequential model (meaning the model consists of a linear stack of layers, so no weird loopbacks and other oddities are included), consisting of several convolution layers with

their corresponding activation layers, along with flatten layers, dense layers and their activation functions, dropout layers, and another activation layer which serves as the classification layer. A full overview of the model is provided in Table 4 in the Appendix.

We evaluate the model using the performance metrics: "accuracy", "precision", "recall" and "F1-Score". A brief explanation of these metrics is provided in Table 1 Below.

Table 1: Performance metrics for the convolutional neural network.

Metric	Description
Accuracy	Correct predictions as a fraction of total predictions
Precision	Relevant items as a fraction of selected items
Recall	Selected items as a fraction of relevant items)
F1-Score	A weighted average of precision and recall

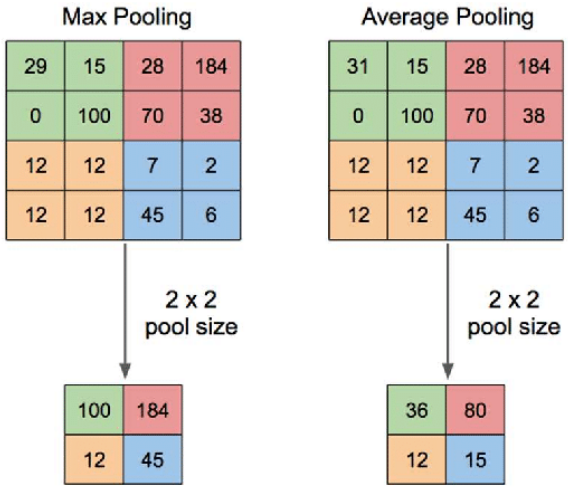


Figure 7: Max pooling layer [10]

5 RESULTS

The Data Set has been created following the methodology, and used to train the two models. The results are as follows:

5.1 SVR

The SVR performed as described in Table 2. Those metrics have been obtained using the scikit-learn library [7], specifically the *metrics.r2_score*, *mean_absolute_error* and *mean_squared_error* functions. The mean squared error has also been plotted against the ground truth value. The mean squared error is lowest between the accident counts [5,10]. It is important to note that 6 outliers ranging from 3000 to 8000 have been left out of this plot for readability reasons.

Table 2: Results of the performance metrics for the SVR.

Performance metric	Result
R2 score on training set	0.4139
R2 score on validation set	-0.0466
Mean Absolute Error	-6.424 (σ0.343)
Mean Squared Error	-111.220 (σ20.203)

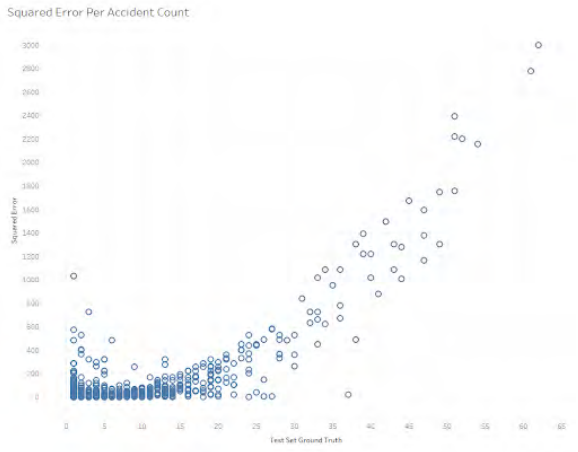


Figure 8: Distribution of squared error relative to real number of accidents per tile.

5.2 CNN

For classification algorithms such as the CNN, we are mainly interested in metrics related to how many labels, and which labels, we got correct. The metrics we use to do this are : accuracy, precision, recall, and f1-score. These metrics can be found in Figure 9.

```
[[  0   0   0 233]
 [  0   0   0 275]
 [  0   0   0 310]
 [  0   0   0 284]]
              precision    recall  f1-score   support

        high       0.00      0.00      0.00       233
         low       0.00      0.00      0.00       275
      medium       0.00      0.00      0.00       310
    very_low       0.26      1.00      0.41       284

    accuracy                           0.26      1102
   macro avg       0.06      0.25      0.10      1102
weighted avg       0.07      0.26      0.11      1102
```

Figure 9: Performance metrics of the CNN.

6 DISCUSSION

This section describes the interpretation of our results,a reflection on the methodology, and opportunities and recommendations for future work.

6.1 Data Preparation and Data Mining

In hindsight, some different choices could be made in the data preparation. Firstly, we could have chosen to limit the geographical scope of data points used in the final Data Set to the Rotterdam area, this could have yielded a better Data Set. Additionally, it might be wise to drop all the points with a value of 1, as these points could also be considered relatively safe. It would be nearly impossible to also include data of "safe" places, as no single Data Set could capture the entire population of accidents, so geographical selection of "safe" data points would be akin to proving purple flamingos do not exist. To experiment with the current methods, different zoom levels could be used in an effort to try and achieve a higher accuracy on the classifiers. There are also some improvements which can be made in the future, which are listed under Future Work.

6.2 CNN

The CNN under performed, and likely was overfit. Inspecting the evaluation report reveals that the CNN indeed picked one class all the time. Interestingly, based on the baseline classes one would expect that the classifier would pick "medium" constantly, however the class picked in accordance to baseline estimation is "very_low". This is most likely due to the train/test division. Another reason why the CNN under performed could be an error in the sample weights, but this should not be a problem as the Data Set is already quite balanced. Additional improvements overall can be found in the Future Works section.

6.3 SVR

In general, the SVR performed better than chance on the training set (Sklearn R2 metric is positive), but performed poorer than chance on a validation set (Sklearn R2 metric is negative). This suggests that there is some statistically significant data available in the image filed, but that the Data Set cannot be used to train a general model. While with a mean absolute error of -6 and a mean squared error of -111 the classifier is far from accurate, the system should be able to give a ball park estimate which can be of value to civil engineers.

In the SVR, the mean squared error is relatively low in the lower regions. That means that if a classification is low, a civil engineer could reasonably assume that a road is relatively safe. This would mean that an improved version of this system could prove to be potentially valuable. However, in this state, our system is not ready for use. While it is an interesting proof-of-concept, there are a number of future steps still to be undertaken.

6.4 Future Work

From our research we conclude that predicting road safety based on map data is possible, but there is still a lot of room for improvement. In this section, we provide some ideas on how our current method can be improved.

Firstly, we can incorporate more features into the classifier. Information such as the number of traffic signs present at a location may help improve performance.Factors like (expected) traffic usage and geographical location could also be valuable features a classifier could use.

Secondly, we can use the same method and apply it to a different Data Set, or an expanded version of the Data Set. The Data Set may contain data from all over the Netherlands (or even more), or include more different types of roads (e.g. include roads which are not highways).

Furthermore, improving the aggregation of data may turn out to be beneficial. Instead of grouping per tile as we do now, the data can be grouped based on road features (such as interchanges, roundabouts, T-junctions, or bridges, among others). Another approach would be aggregation based on k-means clustering. Instead of dividing the map into pre-set tiles, we divide the map in clusters based on where the accidents happen relative to each other. An example of what this would look like is provided in Figure 10 below.

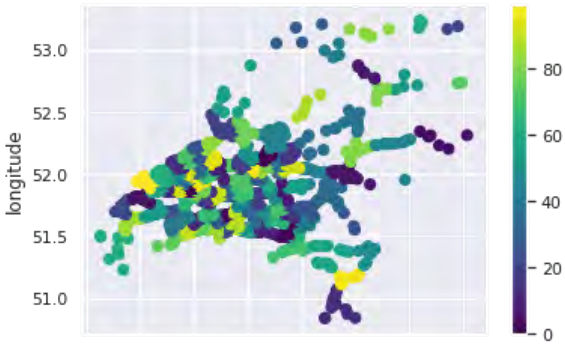


Figure 10: K-means clustering [10].

Moreover, we can use the same Data Set to predict accident types instead of the number of accidents. This way, we get information on what types of accidents would occur, so we can get more information on which parts of the design need to be adjusted.

Lastly, the performance of other types of classifiers can be explored. Our research focuses on two fairly simple classifiers, being the SVR and the CNN, but implementation of other methods such as Long-Short-Term-Memory (LSTM), or other classifiers like Deep Neural Networks (DNN) may yield higher performance than our classifiers do.

7 CONCLUSION

Using accident data, a data mining operation has been conducted, and a Data Set has been collected featuring aggregated location data, accident count, accident types, and risk categories. This Data Set was used to train a Support vector Regression model (SVR) and a Convolutional Neural Network (CNN). The SVR performed in a statistically significant manner, yielding a mean absolute error of -6.424. The CNN reached a 26% accuracy by fitting on the baseline classification.

While this work provides a proof of concept for a novel method to predict the safety of new civil engineering projects before they are built, improvements could, and should be made before applying it on real projects.

ACKNOWLEDGMENTS

We want to thank everyone who contributed to our health and well being, guidance, and help. Specifically we want to thank the creator and maintainers of the OpenStreetMap API [6],the Keras [3] and Smopy[8] libraries for Python. Additionally, we want to thank Mr. B.Bieling from Mapcreator (https://mapcreator.io/) for graciously providing us with maps and interesting comments.

REFERENCES

[1] Md. Zahangir Alom, Tarek Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Nasrin, Mahmudul Hasan, Brian Essen, Abdul Awwal, and Vijayan Asari. 2019. A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics* 8 (03 2019), 292. https://doi.org/10.3390/electronics8030292

[2] P.J. Beryl Princess, S. Silas, and E.B. Rajsingh. 2021. Classification of Road Accidents Using SVM and KNN. *Advances in Intelligent Systems and Computing* 1133 (2021), 27–41. https://doi.org/10.1007/978-981-15-3514-7_3 cited By 0.

[3] Francois Chollet et al. 2015. *Keras.* https://github.com/fchollet/keras

[4] T. Hastie, R. Tibshirani, and J. Friedman. 2013. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer New York. https://books.google.nl/books?id=yPfZBwAAQBAJ

[5] University of Twente. [n.d.]. sb-incidenten-export-rotterdam 1 maart - 31 augustus.zip. http://castle.ewi.utwente.nl/datasciencedata/TRANSPORT/

[6] OpenStreetMap contributors. 2017. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[8] Cyrille Rossant. [n.d.]. rossant/smopy. https://github.com/rossant/smopy

[9] M.V. Valueva, N.N. Nagornov, P.A. Lyakhov, G.V. Valuev, and N.I. Chervyakov. 2020. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation* 177 (2020), 232 – 243. https://doi.org/10.1016/j.matcom.2020.04.031

[10] Muhamad Yani, S Irawan, and M.T. S.T. 2019. Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail. *Journal of Physics: Conference Series* 1201 (05 2019), 012052. https://doi.org/10.1088/1742-6596/1201/1/012052

APPENDIX

Table 3: The Data Set.

Feature	Type
id (Id)	String
versie (Version)	Integer
type (Type)	String
bron (source)	String
starttijd (Starting_time)	Datetime (DD/MM/YYYY HH:MM)
eindtijd (Ending_time)	Datetime (DD/MM/YYYY HH:MM)
situatie_aangemaakt_tijd (Situation_created_time)	Datetime (DD/MM/YYYY HH:MM)
situatie_versie_tijd (Situation_version_time)	Datetime (DD/MM/YYYY HH:MM)
speciaal_geval (Special_case)	String
vild_versie (Vild_version)	String
vild_richting (Vild_direction)	String
vild_primaire_locatie (Vild_primary_location)	Integer
vild_afstand_tot_primaire_locatie (Vild_distance_to_primary_location)	Integer
vild_afstand_tot_secundaire_locatie (Vild_distance_to_secundary_location)	Integer
mobiliteit (Mobility)	String
veiligheids_gerelateerd_bericht (Safety_related_message)	Boolean
primaire_locatie_lengtegraad (Primary_Location_longitude)	Float
primaire_locatie_breedtegraad (Primary_Location_latitude)	Float
secundaire_locatie_lengtegraad (Secondary_Location_longitude)	Float
Secundaire_locatie_breedtegraad (Secondary_Location_latitude)	Float
gedetailleerd_type (Detailed_type)	String

Table 4: Model overview of our convolutional neural network.

Layer	Parameters
Sequential	None
Convolution(Conv2D)	filters = 32, kernel_size = (3, 3)
Activation	ReLu
MaxPooling	pool_size = (2, 2)
Flatten	None
Dense	units = 64
Activation	ReLu
Dropout	rate = 0.5
Dense	units = 4
Activation	Softmax

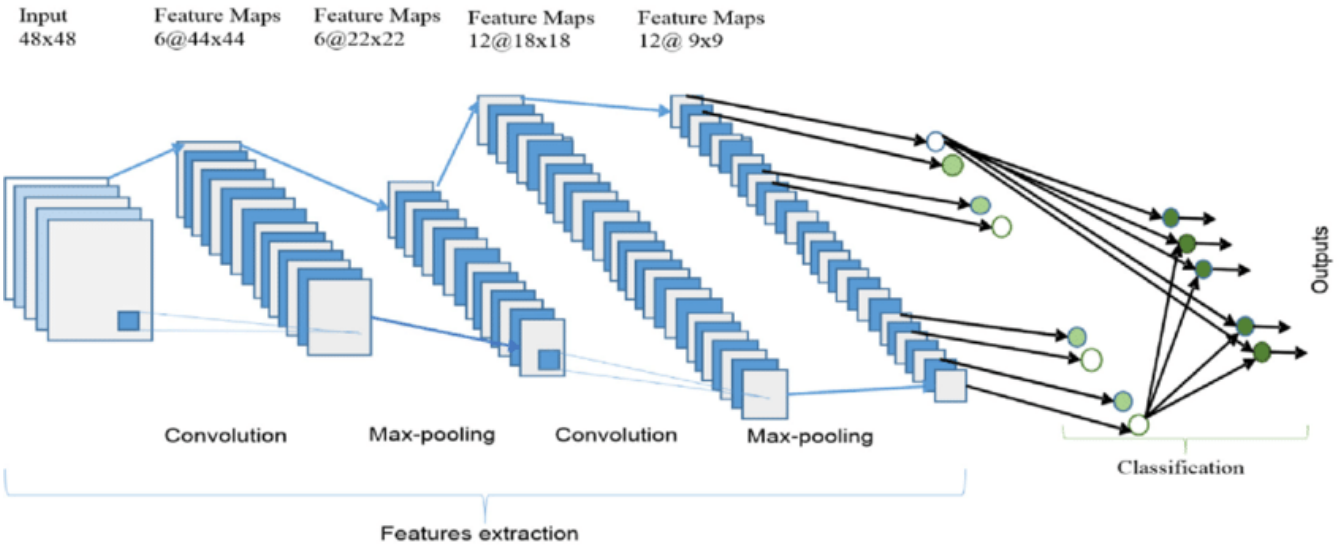


Figure 11: The convolutional neural network [1].