

AUTOMATIC DETECTION OF PLAN SYMBOLS IN RAILWAY EQUIPMENT ENGINEERING USING A MACHINE LEARNING APPROACH

Deian Stoitchkov¹, Peer Breier¹, Martin Slepicka¹, Cengiz Genc², Felix Harmsen²,
Tobias Köhler,² Simon Vilgertshofer¹ and André Borrmann¹

¹Technical University of Munich, Germany

²Signon Deutschland GmbH, Berlin, Germany

Abstract

Exact data in the form of technical drawings and plans of built assets are a significant requirement for the successful operation and reconstruction of such assets. When the consistency between this data and the real world situation cannot be assured, the data is not reliable and needs to be updated by comparing plans and reality. Depending on the size and number of assets this may involve an enormous amount of manual effort. In the scope of this research, an approach for supporting and automating such a process by utilizing concepts developed in the field of machine learning was developed. This paper focuses on the interpretation of technical drawings in terms of detecting and classifying plan symbols as this is a time intensive and error prone process when done manually. It is described how the capabilities of Convolutional Neural Networks are employed in analyzing images to automatically detect important plan symbols in the field of Train Traffic Control and Supervision Systems and how those networks are trained without the need for a time consuming-manual labeling process.

Introduction

The railway network in Germany, which consists of more than 30.000 kilometers of railway, is mainly operated and managed by the DB Netz AG. As this integral part of German infrastructure has grown over the last century both tracks and railway equipment are of various technological levels. This is also the case for corresponding technical drawings and plans. However, management, operation and especially rebuilding and reconstruction of railway infrastructure rely heavily on the availability of such technical drawings which need to be exact and up to date. In a vast railway network of several thousand kilometers that has grown over decades, discrepancies between archived drawings and the actually built infrastructure are almost inevitable due to various circumstances. As the purely manual effort required for comparing and updating the stock data consisting of technical drawings and plans involves an enormous amount of manual effort, we devel-

oped an approach for supporting and automating parts of this process by utilizing concepts developed in the field of machine learning.

The research presented here is part of the RIMcomb research project (Railway Information Modeling: Equipment technology for rail infrastructures (Vilgertshofer et al. 2018)). Among the main goals of the research project is the digitization of the information on conventional 2D-drawings depicting railway equipment. While most drawings are available as digitally stored images by now, the interpretation of these plans and, most importantly, the various symbols they contain has to be undertaken manually to create a semantically rich digital railway representation.

Our approach aims at supporting this process in order to reduce the manual effort by automating at least parts of this image interpretation process. The first step towards this goal is the automatic recognition and highlighting of plan symbols on a given drawing and the subsequent storing of their count and location. To achieve this, we employed Convolutional Neural Networks (CNN) and developed a process in which to train these networks without the need for a time-consuming manual labeling process to generate training and testing data. The results can then be compared to the actual situation on the track in order to check for inconsistencies or confirm the accuracy of a plan. Naturally, this comparison process also requires the recognition of infrastructure elements on the track as a counterpart. In the scope of the RIMcomb research project, a second approach for automating this process has been developed and is summarized in the following section (Genc et al. 2018). The combination of those approaches is currently underway and will result in a semi-automated process of the otherwise highly labor intensive comparison of the as-planned and as-built situation of railway infrastructure.

This paper is structured as follows: In Section 2 we will discuss existing approaches to detect and classify symbols in technical drawings. We will further describe the technological background that our approach involves. Additionally, we will give an overview of a related development that focuses on the recognition of railway equipment in

images/videos and acts as a counterpart to our approach. Section 3 provides a detailed description of our methodical approach and the conducted research work, while Section 4 focuses on our implementation work and the general quality of our results. In Section 5 we show how we apply our approach on a set of real-world technical drawings to deliver a proof of concept. Finally, we will sum up our findings and describe how we will further develop our approach.

Related research and theoretical background

Symbol recognition in technical drawings is a well-known challenge in the engineering field. In technical drawings, symbols often appear to be strongly distorted or overlapped with other objects, making their recognition a significant challenge. Therefore, numerous solutions to this problem have already been proposed.

Luqman et al. (2009) describe a method which represents symbols by their graph-based signatures and a Bayesian network is trained to encode the common probability distribution of symbol signatures. The Bayesian network is trained in two phases – the structure learning phase and the parameter learning phase. Another approach is described by Weber & Tabbone (2012), where a template matching operator HMTAIO (Hit or Miss Transform Adapted to Information Overlapping) is used. The advantage of this approach is its robustness against occlusion and overlapping. Another interesting approach for symbol spotting is proposed by Rusiñol & Lladós (2005). The authors describe their approach as follows: “In this paper, we present a method to determine which symbols are probable to be found in technical drawings using vectorial signatures. These signatures are formulated in terms of geometric and structural constraints between segments, as parallelisms, straight angles, etc.” (Rusiñol & Lladós 2005). Different symbol recognition methods are also described by Le Bodic et al. (2012) and Nayef et al. (2012).

Although many approaches have been proposed, they could not deliver satisfactory results for detecting railway equipment symbols. In our approach, we train artificial neural networks to recognize symbols in engineering plans. This method shows promising results and can be used in various contexts. It is not only invariant of scale and rotation operations, but it also works for heavily distorted symbols. The approach recognizes many types of symbols, thus demonstrating its practical applicability.

Convolutional Neural Networks (CNNs)

Artificial neural networks are used for different tasks such as text, speech and image recognition, and even self-driving cars. Convolutional neural networks (CNNs) have

been used here, as they appear to be very useful for image recognition. One of the first CNNs developed is the LeNet (LeCun et al. 1998), and since its creation, there has been a tremendous improvement in the field of image recognition and object recognition in general.

CNNs use filters to detect patterns in an image. An image is searched for these patterns, and a value is saved, which represents how good each pattern matches the images at specific locations. In this way, an image can be inspected for features such as horizontal or vertical lines. Via combination, they can represent more complex shapes such as symbols. Figure 1 shows an example for two filters of a horizontal and a vertical line on the left side and how they are matched in the image on the right-hand side.

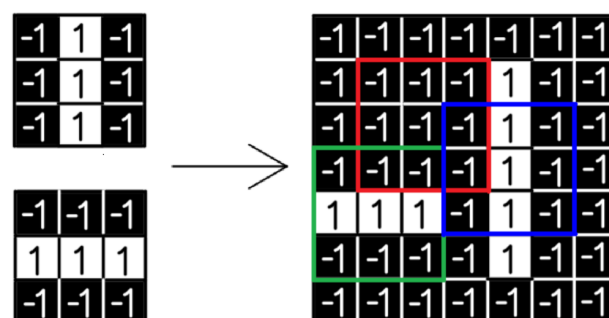


Figure 1: Searching for filters in an image

Each filter pixel is multiplied with the corresponding pixel from the image, and the resulting values are added up. Afterward, this value is divided by the pixel number of the filter. If the filter is found in the image with a 100% accuracy, the result for this location of the image is 1 (upper filter with the blue box in the image). On the other hand, if the filter has no common values with the current location in the image, the result is -1. The overall result is a map which represents where the different features occur in the image. A ReLU activation function can then be used to normalize the values in the feature map that was calculated, which improves the training process (Figure 2).

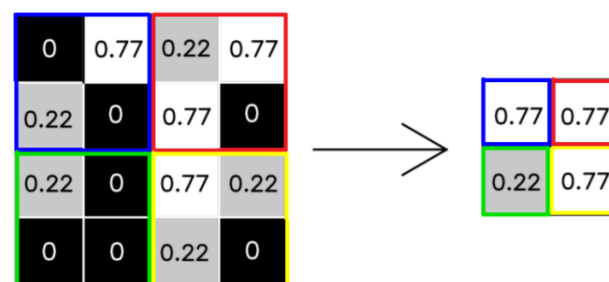


Figure 2: Max pooling

Images usually consist of thousands or even millions of pixels, which can make the training of a CNN computationally expensive, as bigger images result in a bigger feature map. Therefore, an operation is introduced that reduces the size of the feature map. We chose the so-called max pooling operation. It takes only the maximum value of a given window and saves it at the correct location in a new, smaller feature map (Figure 2). These techniques can be used multiple times in so-called layers, where each layer takes as input the output of the previous layer. The image gets more filtered for each convolution layer and smaller for each pooling layer (Stoitchkov 2018).

The extracted features are then fed into the neural network. A very simplified example neural network can be seen in Figure 3. By adjusting the weights (marked with a W in Figure 3) between the neurons, an output is calculated from a given input. In the training process, the neural

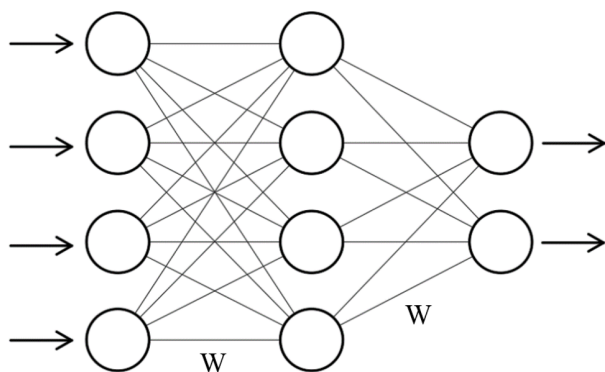


Figure 3: Neural network structure

network tries to minimize the difference between the output it has generated and the ground truth result which is targeted. In order to reach a good accuracy, meaning exact recognition, thousands of images are required. However, one of the biggest challenges of training a neural network is to prevent overfitting. Overfitting occurs when the training images are similar to one another, and the network is extremely optimized in recognizing these images but fails to recognize a new set of images. Instead of learning, the model then starts to memorize the training data.

Recognition of infrastructure elements in video data

As already mentioned Genc et al. (2018) have developed an approach that also relies on the use of CNNs in order to automate the process of detecting and cataloging infrastructure elements on railway tracks. In contrast to the method presented in this paper, their approach focuses on video data, that was obtained by filming the tracks with a camera positioned at the front of a train. While these videos have been used to manually detect existing infras-

tructure elements in the past as a basis for maintenance or reconstruction tasks, they are now used as the basis for an automated process. Figure 4 gives an impression of the types of objects that are recognized (ETCS balises, light signals, and switches) and the general results of the approach. The findings in this research can now be used to analyze infrastructure data to save cost and time compared to a manual process.

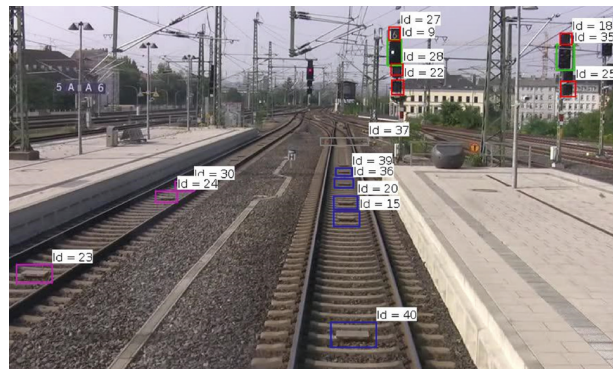


Figure 4: Application of object recognition (Genc et al. 2018)

Methodical Approach

The project NNdips (Neural Network for the detection and interpretation of plan symbols) aims to facilitate the detection of plan symbols for large amounts of technical drawings, i.e., plans of train tracks. The process is composed of three main parts, namely the import and preprocessing of data, the training of CNNs and the detection of symbols using the CNNs. All steps have been automated and thus provide a technique that does not require classification by users. We were provided with a large data set of real technical drawings and plans of railway tracks by the DB Netz AG. These formed the basis of our research in terms of data and are especially important as they were used for validating our results.

In the first stage of this project, the framework for the research was set. As there are numerous ways to utilize neural networks, it was decided at first which type of neural network was to be used and how it should be applied to the data. Considering that the aim was to detect plan symbols in railway plans – usually image files – we decided on using CNNs. As mentioned before, CNNs are particularly suitable for image recognition, and their use has been successfully tested for symbol recognition purposes by Stoitchkov (2018).

The symbols that need to be recognized are tiny compared to the complete plan and do not occur in large numbers.

Also, on most technical drawings there is much white space where it is unreasonable to scan for a symbol as nothing is depicted. Therefore, it makes sense to “cut” the plan into small pieces, so-called regions of interest (ROI) and to scan only those ROIs, which are not empty. In other words, white ROIs are ignored to minimize the amount of data and thereby the computational effort.

The size of those ROIs is a critical parameter, as the computational effort increases disproportionately when very small ROIs are used. In the case of large ROIs, the training time increases drastically while at the same time the detection accuracy decreases. At the same time, it is crucial how the plan is divided into regions. It is possible that a significant symbol is located on a border between two regions of interest and thus will not be detected, as the partial symbol cannot be recognized by the CNN in either of the two regions. We found that the simplest solution for this problem is an overlap of the ROIs at the expense of some computational overhead.

Generation of training Data

Providing thousands of input images for the training process is usually a big challenge and can result in tedious work if the images have to be selected and classified by hand. In the case of plan symbol detection, training images can be generated artificially due to the inherently repetitive nature of drawing symbols. This generation process is done in the following steps.

In a first attempt, a large background image was created by lumping some random image parts together, such as parts of technical drawings, text elements, and random patterns. Subsequently, thousands of smaller images containing the symbol to be detected and having the same size as the ROIs in the recognition process were created. This was done by copying random parts of the same size from the background image and placed the symbol on each of them. To ensure that a large variety of images was generated, this process was entirely randomized by choosing arbitrary parameters, such as the position and rotation of the symbol relative to the image (cf. Figure 5).

After many tests, which did not yield sufficiently accurate prediction results, we concluded that an improvement of the generation process was necessary. A first attempt adapted the background image and made sure it mostly contains elements such as line shapes, numbers, and text that may occur in the technical drawings that should be scanned later on. The random patterns, which were put in the background image before, did not provide any increase in training accuracy. The plan elements (such as line shapes), however, helped to reduce the detection of false positives drastically.

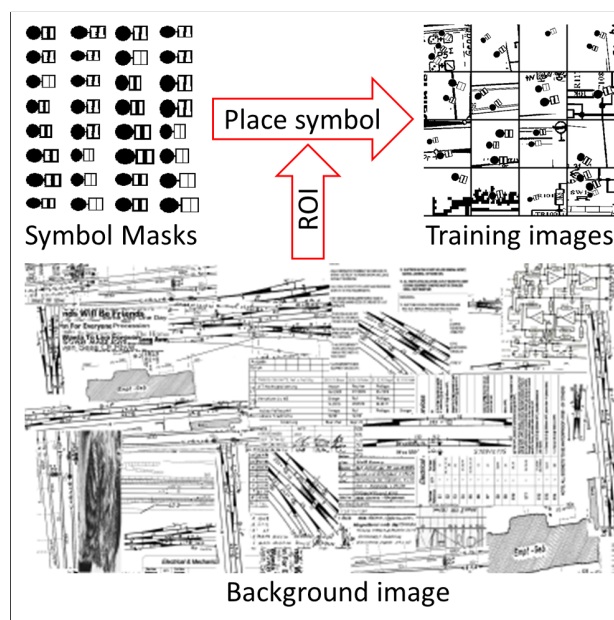


Figure 5: Cutting the background image into ROIs and placing training symbols in different configurations

Afterward, the symbol placement was also improved. For this, a set of variations of the symbol the CNNs was supposed to recognize was created. This was achieved by resizing and stretching the symbol. These measures increase the detection capabilities of the trained CNNs and therefore reduce the number of false negatives.

For every symbol that the network is being trained for, the same background image may be used. However, to improve the training accuracy, the background image was modified by placing plan sections containing markings similar to the target symbol to reduce false positives and increase the specificity of the resulting classifier.

At the same time, training images were also created to generate the output data which contains the exact information that the CNNs are supposed to predict. For each training image, the information is stored separately.

Training process

Before the training process could be initiated, it is necessary to compile a CNN model. To create this CNN model various parameters have to be adjusted, such as the amount and type of layers, the number of filter types, the mask size of the filter (kernel size), the type of activation function, loss function (error calculation) and optimizer.

As there is no single perfect solution to set up the model, the best parameters had to be determined by systematic experimentation. In this regard, we created different model

setups by changing one parameter per setup, trained and tested them for accuracy. The models with the best results were then further refined until we obtained a model setup with every good recognition capabilities. In each of these refinement stages, we trained with only a reduced amount (a few hundred) of input images, as this process otherwise would have been inappropriately time-consuming.

After that, we trained a CNN with the obtained model setup and a few thousand input images and applied it on real plans of train tracks. The results of these scans revealed that the model still could not match the expectations, as it failed to detect symbols which are located close to each other.

To further increase the detection accuracy, the CNN was split into three more specialized CNNs. The first CNN is supposed to detect how many symbols are located within the detection range (Region of interest) and then to initialize one of the other CNNs, which are responsible for predicting the location of the symbol center of one or two symbols (cf. Figure 6).

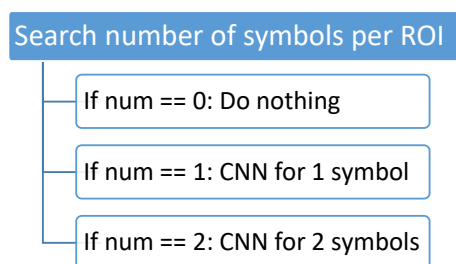


Figure 6: CNN types

In a final refining process, we changed the loss function of the number recognition CNN to a classifying method rather than a mean square error evaluation.

Overfitting

Because the training images are generated artificially, there is a high chance for overfitting. Therefore, two types of input symbols are required. The first type (the training symbol) is a set of up to five images depicting different variants of the symbol we want to detect. With these images, we generate the training data set. The other testing symbol is cut out from a real, presumably slightly distorted plan and we generate another set, also with a different background image – the testing data set. From the training data set we take 15% as validation data, which is also used for calculating the accuracy. We measure the accuracy for both the validation and testing data set and take the average as our final accuracy.

In the training process, we train one epoch, pause, apply the trained neural network on both image sets, measure the

accuracy and save the neural network only if it yields the best accuracy compared to the prior epochs. If the neural network overfits over the epochs, the accuracy of the test data will decrease and, because of the averaging with the training data set, the overfitted network will be discarded.

Implementation and results of the training process

Facilitating the training and recognition process

We implemented the derived algorithms into a GUI (Graphical User Interface) in order to simplify their usage. It allows for importing plans in several formats, such as .pdf, .png or .tif and the detection of pre-trained symbols in the imported plans. Positions and other results are visualized and can be exported as text files or as annotations at the positions of detected symbols in PDF-files. Another feature is the option of training the necessary CNNs automatically for new symbols. This can be done by using the predefined parameters or using advanced options that control, e.g. ROI size, number of epochs or kernel size.

Provided that the user has already trained at least one set of CNNs, the workflow of our software is straight forward. The technical drawings that should be evaluated are imported, then assigned an ID and displayed for the user to check if any mistakes happened in the conversion process. After that, the plan is scanned for a chosen symbol by using the respective set of CNN. The results of the scan are then presented on the same screen in a list box with an ID and their respective position in pixel coordinates. All results can be selected one by one, and on selection, their position on the plan will be visualized with crosshairs as well as a zoomed image. At this stage, the user can manually delete falsely detected symbols and check the overall quality of the scan. Finally, the user can export the obtained and reviewed results. Figure 7 illustrates the workflow schematically.

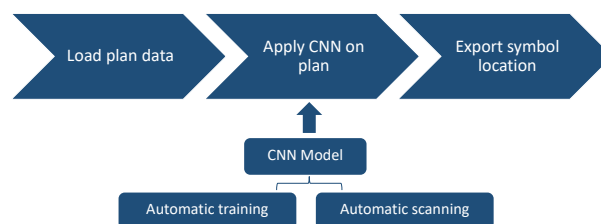


Figure 7: NNdips workflow

Used software

To build the CNNs we chose the TensorFlow¹ framework for Python. TensorFlow is an open-source software library for dataflow and contains a symbolic math library. It is frequently used for building neural networks and is released under the Apache 2.0 open-source license. To enable fast testing, the Keras API² was used in this project, which is an API that runs inside TensorFlow. We avoided conflicts with the compatibility of different programming languages by writing the GUI with the TkInter³ library, which is the standard GUI in Python.

Accuracy

The accuracy is measured in two different ways. For the number of symbols, it is measured in percent, where the recognized number is compared to the ground truth number of symbols in the ROI. It returns 1 for matching numbers or else 0 and the sum divided by the total number of images gives the percentage of the correctly predicted number of symbols. On the other hand, the accuracy for the location of a symbol is measured as a distance in pixels, where the distance denotes the deviation from the recognized coordinates to the ground truth coordinates of a given symbol.

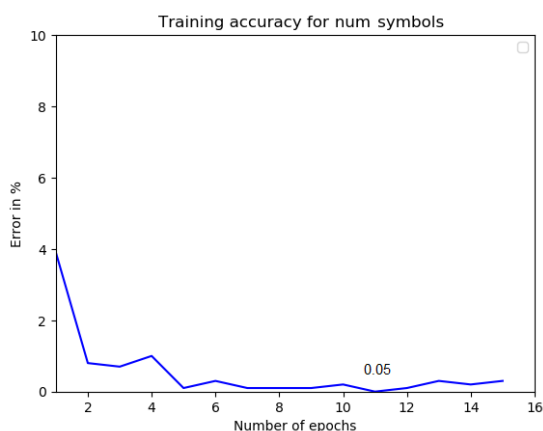


Figure 8: Accuracy for the number of symbols

Figure 8 shows the accuracy of the detected number of symbols, and it can be seen that the network learns very fast for this specific task. Figure 9 displays the learning curve for the location of one symbol. Here the learning happens slower and more time is required for accurate results. We explain this by the complexity of the problem, since deciding between 0, 1 or 2 is an easier task than to find the exact x and y coordinates in a ROI. Both figures were

¹<https://www.tensorflow.org/>

²<https://github.com/keras-team/keras>

³<https://wiki.python.org/moin/TkInter>

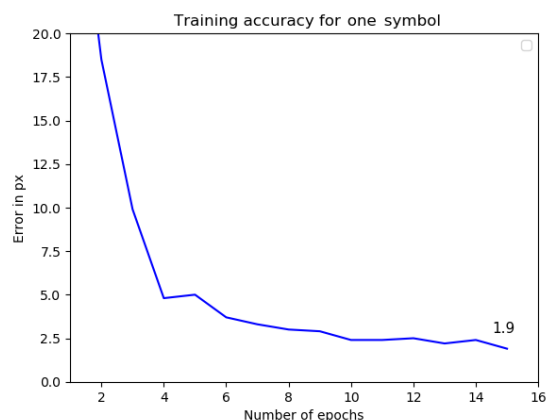


Figure 9: Accuracy for one symbol

created in the scope of the training process with the default values of the ROI size (160x160 pixels) for 15 epochs and an amount of 23.996 training images, 4.235 validation images, and 4028 testing images. Tests were conducted on different symbols, and the accuracy was consistently above 99% for the recognition of the number of symbols in a ROI and less than 2 respectively 4 pixels deviation for one respectively two symbols. However, when the neural network is applied on a plan, there are thousands of ROIs to be searched and the error multiples with the number of ROIs.

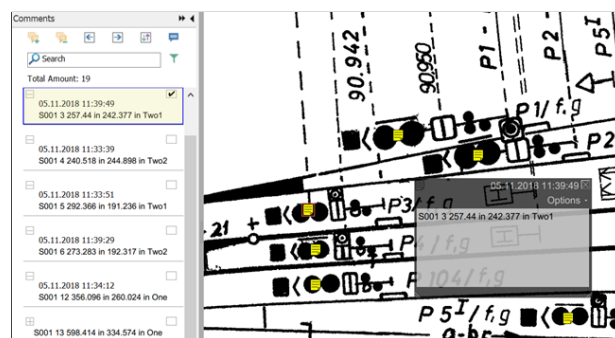


Figure 10: Visualization of the results in a PDF file

Proof of concept and use-case scenario

The output of the CNNs is the coordinates for each found symbol. These can be visualized by marking the position in the associated plan. Figure 10 shows an example for the output in a PDF file. The positions are marked as annotations. Another possibility is the export of a TXT file. We expect users to supervise the outputs and allow deleting and commenting on single results in the GUI.

So far we tested our networks with over 25 different plans












	Plans	manual count	Nndips	doubles	false +	false - [%]
	18	215	366 (197)	169	9	12,56 (27)
	1	4	7 (4)	3	0	0 (0)
	4	22	55 (35)	20	16	13,64 (3)
	4	13	42 (37)	5	25	7,69 (1)
	4	37	70 (34)	36	2	13,51 (5)
	10	28	51 (30)	21	2	0 (0)
	6	16	24 (14)	10	0	12,50 (2)
	4	96	156 (90)	66	0	6,25 (6)
	7	20	43 (21)	22	1	0 (0)
	22	29	74 (49)	25	20	0 (0)
	26	43	133 (76)	57	36	6,98 (3)

Figure 11: Results of the verification process

provided by the DB Netz AG. In those plans, we selected more than ten symbols to verify our previously described method. Figure 11 summarizes the results of this verification process. Each line shows the actual number of symbols present in a set of plans (column *manual count*). The column *Nndips* shows the number of hits that the CNN produced. Since we usually have multiple hits for the same symbol, the number in parenthesis shows the actual number of symbols found - without counting the multiple hits. The number of multiple hits is shown in the column *doubles*.

In some cases the CNN produced a falsely positive result (a hit at a position where no symbol is present) - the respective number is given in the column *false +*. Unfortunately, the CNN does not recognize a symbol at all times. Such false negatives are shown in the column *false -* as percent values (the actual number of symbol which were not recognized is given in parenthesis; the percent value is calculated by dividing the number of false negatives by the number of actually present symbols, e.g., $\frac{27}{215} = 12,56$ in the first row).

While those results are not perfect, they nonetheless indicate that our approach shows promising possibilities - especially in regard to the very short time required for training the CNNs (see below). We are currently working on further improving the general accuracy and also on automatically removing the unnecessary multiple hits.

Challenges

Training CNNs for the detection of symbols in technical plans pose multiple challenges. Previous knowledge about the occurrence of the symbols is essential. In the given technical plans, we found the occurrence of multiple symbols closely together. One issue that results is the multiple detection of the same symbol in various ROIs. This is a well-known issue especially in object detection in videos. Other works such as Breuers et al. (2016) tackle this by deleting multiple hits by calculating an overlap-area-ratio of bounding boxes around the detected objects. We did not apply any of the possible techniques here, as the plan data yields cases where multiple symbols are incredibly close together, and a minimal distance approach or a bounding box overlap could delete distinct hits. Further investigation of this phenomenon could minimize the amount of multiple detection of symbols.

Another challenge is the rare case of more than two symbols in a single ROI that are not detected by the overlap of ROIs. We built a fourth CNN for the task of the detection of three symbols in a ROI but found lack of accuracy. The fourth CNN worsened the amount of multiple found symbols, referred to as false positives. If in further testing, symbols are overlooked by the algorithm due to more than two symbols in a ROI, this error can presumably be corrected by merely reducing the ROI size.

Time of training and recognition

When using the parameters and symbols shown in this paper, the time for the training process of a single symbol took between 20 to 30 minutes on a Computer with an 8 core i7-7700 CPU at 3.60GHz and an NVIDIA Quadro P2000 graphics card. The recognition process, which consists of the calling of the trained model and drawing of the symbols on a single plan took about ten seconds on the same machine. Similar results could be achieved on a laptop with an NVIDIA GeForce 940MX; thus the training can presumably be done within a reasonable time on any modern graphics card.

Summary and outlook

This paper has introduced a method to automatically recognize important symbols in technical drawings of railway infrastructure by using a Convolutional Neural Network. The authors have shown how their approach was developed, implemented and tested in order to reduce the manual effort that would otherwise be necessary in order to obtain digital information from the aforementioned technical drawings. This information can then be further used as the basis for planning purposes or to check the accuracy of the drawings against the actual as-built situation. Since the accuracy of the method is only at approx. 90 % at the moment we are continuously improving the automated training process. In regard of the fact that the training process does not require a large number of manually labeled images and is automated as well as fast, we conclude that the accuracy of the approach shows the possibility of reducing the manual effort in recognizing plan symbols. Furthermore, we plan to combine our method with the approach by Genc et al. (2018) to enable a semi-automated consistency check of technical drawings. For this purpose, we are currently acquiring video data as well as plan data for a specific stretch of railway track. By applying both approaches to the data and comparing the results we plan to show the applicability of the combined approach.

Acknowledgements

We gratefully acknowledge the support of the Bavarian Research Foundation for funding the project.

References

- Breuers, S., Yang, S., Mathias, M. & Leibe, B. (2016). Exploring bounding box context for multi-object tracker fusion. In: *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, pp. 1–8.
- Genc, C., Harmsen, F. & Köhler, T. (2018). Automated recognition of infrastructure elements using neural networks. *SIGNAL+DRAHT | Ausgabe 009/2018*.
- Le Bodic, P., Héroux, P., Adam, S. & Lecourtier, Y. (2012). An integer linear program for substitution-tolerant sub-graph isomorphism and its use for symbol spotting in technical drawings. *Pattern Recognition* 45(12), pp. 4214–4224.
- LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE*, Vol. 86, IEEE, pp. 2278–2324.
- Luqman, M. M., Brouard, T. & Ramel, J.-Y. (2009). Graphic symbol recognition using graph based signature and bayesian network classifier. In: *10th International Conference on Document Analysis and Recognition*, IEEE, pp. 1325–1329.
- Nayef, N., Afzal, M. Z. & Breuel, T. M. (2012). Learning feature weights of symbols, with application to symbol spotting. In: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR)*, IEEE, pp. 2371–2374.
- Rusiñol, M. & Lladós, J. (2005). Symbol spotting in technical drawings using vectorial signatures. In: *International Workshop on Graphics Recognition*, Springer, pp. 35–46.
- Stoitchkov, D. (2018). Analysis of methods for automated symbol recognition in technical drawings. *Bachelor's thesis*, Technical University of Munich
- Vilgertshofer, S., Stoitchkov, D., Esser, S., Borrmann, A., Muhič, S. & Winkelbauer, T. (2018). The RIMcomb research project: Towards the application of building information modeling in railway equipment engineering. In: *Proceedings of the 12th ECPPM*, Copenhagen, Denmark.
- Weber, J. & Tabbone, S. (2012). Symbol spotting for technical documents: An efficient template-matching approach. In: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR)*, IEEE, pp. 669–672.