Machine learning and computer vision for urban drainage inspections

Meijer, D.W.J.

4 Convolutional Neural Network Classification

The process of CCTV sewer pipe inspections is both labour-intensive and error-prone. Other researchers have suggested machine learning techniques to (partially) automate the human review of this footage, but the automated classifiers are often validated in artifial testing setups, leading to biased results that do not translate well to practice.

In this chapter, we design a convolutional neural network (CNN) and apply this validation methodology to automatically detect the twelve most common defect types in a dataset of over 2 million CCTV images. We also discuss suitable evaluation metrics for this specific classification task — most notably 'specificity at sensitivity' and 'precision at recall' — and the importance of using a validation setup that includes a realistic ratio of images with defects to images without defects, and a sufficiently large dataset. We also introduce *'leave-two-inspections-out'* cross validation, designed to eliminate a data leakage bias that would otherwise cause an overestimation of classifier performance.

With this dataset and our validation methodology, our CNN outperforms the state-of-the-art. Classification performance was highest for intruding and defective connections and lowest for porous pipes. While the CNN is not capable of fully automated classification at sufficient performance levels, we determined that if we augment the human operator with the CNN, this may reduce the required human labour by up to 60.5%.

4.1 Introduction

In this chapter a possible method to automate the inspection process is demonstrated and shown to be viable. While the performance of the method is noteworthy, we consider the most important contribution of this chapter not to be this method itself, but rather the methodology used to validate these results and assess their impact if used in practice.

4.1.1 Image Classification

Image classification is the primary way in which we attempt to address the automation of the inspection process. This classification assumes that we have *training data*, consisting of a set of images of CCTV footage, each of which has an assigned *label*, which indicates whether specific types of defects are present and visible in the image. The classifier infers a statistical relation between the images and the labels, which allows it to make predictions about the labels of images that we do not know the true labels for, such as recently recorded images that still require assessment.

Traditionally, the automated classification of images is done with extracted image features [1], which are known to capture information that is less visible in raw pixel values. Recently, this approach has been mostly replaced by convolutional neural networks [2] (CNNs, explained in more detail in section 2.3). CNNs employ *end-to-end* learning: the original pixel values are used as inputs, and the CNN learns the feature extractions as well as how these features relate to the labels. This allows for extracted image features that are more specialised to the classification task. There is one main downside to this approach: there are a lot more parameters to fit, as the extracted features also need to be inferred from the image data. Two resulting limitations are that a lot more data is required to fit all these parameters,

[1] Szeliski, R. 2010. *Computer Vision: Algorithms and Applications*, 1st ed. Springer-Verlag, Berlin, Heidelberg

[2] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C., and Li, F. 2014. Imagenet large scale visual recognition challenge. *CoRR abs/1409.0575*

and hyperparameter optimization becomes more difficult as the hyperparameter search space (how many filters and of what shape) increases drastically compared to traditional methods.

The impact of the data availability problem can be lessened with *transfer learning*, by using a network that has been pre-trained on a different set of images [3], but then we may also reduce the benefit that the CNN may have in training the convolutional filters specifically to the data and the task at hand. Still, this approach is often favored over a random initialization of the network parameters to save time [4].

[3] Hoo-Chang, S., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D., and Summers, R. M. 2016. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging 35*, 5, 1285

[4] Oquab, M., Bottou, L., Laptev, I., and Sivic, J. 2014. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1717–1724

4.1.2 Classification Result Validation

To assess the performance of a trained classifier, we need a test set that is independent of the training set. To use (part of) the same training set as the test set introduces a bias, which means we are not measuring how well the classifier performs, but only how well it can recognise before-seen data. Since two independent data sets may be difficult to come by, often a portion of the training set is set apart to be used as the test set. [5] The training and test set are not independent in such a scenario and likely contain the same sampling bias, but it is often the best we can do.

To assess the performance accurately, some variance in the samples in the test set is required, which means many samples are required, and a significant portion of the training set may have to be set apart. A significant reduction in size of the training set could itself impact the performance negatively, leading us to underestimate the actual performance of the classifier due to lack of training data. An often used technique to circumvent this problem is *k*-fold cross validation, as outlined in section 2.1.3.

[5] Bishop, C. M. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg

Besides a test set, the performance metrics have to be defined. The most common performance metric used for classification is the *accuracy*, the percentage of correctly classified samples. However, the performance metric should be chosen based on the task at hand, and accuracy is not a good choice for unbalanced classification problems, such as this particular problem, as it favors correct classification of the majority class. [6] Most performance metrics can be thought of as some function of the false positive rate (FPR) and the false negative rate (FNR). A classifier can often be tuned after it has been trained, making it essentially a family of classifiers. In such cases the performance may change as a function of this tuning, and it can be worthwhile to use performance metrics that are independent of which member of the family of classifiers is used. Examples of such metrics are the receiver operating characteristic, or the Pareto-boundary of any combination of metrics [7].

[6] further discussion on this can be found in section 2.1.4

[7] Bishop, C. M. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg

4.1.3 Related Work

Researchers have already applied machine learning techniques to the task of automating sewer inspections. But realistic validation of such methods is often of less note in such articles. As actual defect rates are often very low, in the order of magnitude of 1% of images captured by CCTV — in our dataset we found 0.8% images with defects — it is curious that many authors test their methods on artificial test sets that contain 50% defects. We feel that such a result might be interesting in a vacuum, but gives no indication of the actual 'real-world performance' of a classifier. A relevant selection of research is discussed in this section.

Chae and Abraham [8] use a (non-convolutional) neural network to learn various attributes in relation to the existence and severity of cracks from images of the inner surface of sewer pipes. Their neural network is trained on 20 images

[8] Chae, M. J. and Abraham, D. M. 2001. Neuro-fuzzy approaches for sanitary sewer pipeline condition assessment. *Journal of Computing in Civil engineering 15*, 1, 4–14

and tested on 13 images, so the actual applicability remains unclear.

Yang and Su [9] compare two SVM approaches and a neural network, trained on wavelet filter responses of images. The classifiers were only applied to images containing defects, and subsequently used to classify *what* defect was present in the image. This means no information is available regarding the false detections in images without defects.

Guo et al. [10] use image registration (alignment of pixel locations) and the absolute pixelwise difference between images to classify image regions as defective or healthy. The method is tested on a dataset consisting of 51 images of defective pipes and 52 images of healthy pipes, with reported accuracy and false alarm rates.

Halfawy and Hengmeechai [11] present an algorithm for crack detection in CCTV inspections, based on a Sobel filter and morphological operations. As the model is partially based on expert knowledge, it does not require a large dataset to train, and it was tested on a dataset with 50 images containing cracks and 50 images not containing cracks.

Halfawy and Hengmeechai [12] improve on their previous work, now training an SVM with varying kernels with HOG features extracted from CCTV images, and report more meaningful performance metrics such as precision and AUROC. The experiments are still performed on a test set that consists of 50% images with defects, so it still tells us very little about real-world performance.

Kumar et al. [13] are one of the first to use convolutional neural networks to exploit end-to-end learning in sewer CCTV defect detection. They focus on three different defect types and train the network three times, once for each defect. They also report the precision as one of their performance metrics and use a training set consisting of 12,000 images, but their test sets also consist of 50% images with defects, again limiting their obtained results to such an artificial scenario. In our work, we reimplemented their sug-

[9] Yang, M.-D. and Su, T.-C. 2008. Automated diagnosis of sewer pipe defects based on machine learning approaches. *Expert Systems with Applications 35,* 3, 1327–1337

[10] Guo, W., Soibelman, L., and Garrett Jr, J. 2009. Automated defect detection for sewer pipeline inspection and condition assessment. *Automation in Construction 18,* 5, 587–596

[11] Halfawy, M. R. and Hengmeechai, J. 2013. Efficient algorithm for crack detection in sewer images from closed-circuit television inspections. *Journal of Infrastructure Systems 20,* 2, 04013014

[12] Halfawy, M. R. and Hengmeechai, J. 2014. Automated defect detection in sewer closed circuit television images using histograms of oriented gradients and support vector machine. *Automation in Construction 38*, 1–13

[13] Kumar, S. S., Abraham, D. M., Jahanshahi, M. R., Iseley, T., and Starr, J. 2018. Automated defect classification in sewer closed circuit television inspections using deep convolutional neural networks. *Automation in Construction 91*, 273–283

gested convolutional neural network and performed tests on our dataset, which more accurately represents a real-world scenario.

Myrans et al. [14] train an SVM and a random forest on extracted GIST features from CCTV images. They use 25-fold cross validation and provide the ROC curve along with the misclassification rates for various defect types, but unfortunately work with a dataset that consists of approximately 37% images with defects, which is not representative of a realistic scenario.

[14] Myrans, J., Everson, R., and Kapelan, Z. 2018. Automated detection of faults in sewers using cctv image sequences. *Automation in Construction 95*, 64–71

In later work, Myrans et al. [15] combine both the SVM and the random forest on a dataset in which 'approximately half' the images contained defects, and obtain results superior to either individual classifier. Again, unfortunately the validation results are not representative of a real-world scenario because of the high prevalence of defects in the test set.

[15] Myrans, J., Kapelan, Z., and Everson, R. 2018a. Combining classifiers to detect faults in wastewater networks. *Water Science and Technology 77*, 9, 2184–2189

4.2 Data Exploration

A dataset has been kindly provided to us by Dutch sewer inspection company *vandervalk+degroot*. The data has two components: the images themselves, and the accompanying inspection reports. The data encompasses 30 inspections from 11 Dutch municipalities, for a total of 2,202,582 images from 3,350 different concrete pipes ranging in diameter between 300 mm and 1000 mm.

4.2.1 Image data

The images have been collected with the RapidView IBAK Panoramo® pipeline inspection system [16]. While the Panoramo software can be used to inspect the pipe in a virtual 3D environment, for this study we merely used the 2D images used

[16] IBAK Helmut Hunger GmbH & Co. KG. 2015. Panoramo® 3d optical pipeline scanner. `http://www.rapidview.com/panoramo_pipeline.html`. Accessed: 2018-12-05

to create these 3D environments as input. The Panoramo system does not record video, but still images with a strobe light, spaced 50 mm apart. This allows for improved image quality, without the need to stop the inspection vehicle from moving. The system is equipped with a front-facing and back-facing camera, each with a 185° wide angle lens. The images from the back-facing camera are slightly occluded by parts of the inspection vehicle and the chain that lowered it into the pipe, so our dataset contains only the images from the front-facing camera.

The images are in 24-bit RGB format, 1040×1040 pixels, JPEG images. No information was given about the compression level, but the images range from 19 KiB to 447 KiB. Four randomly selected sample images are shown in Figure 4.1.

An important feature of the images recorded by the Panoramo system is that the images are spatially aligned. After the device is lowered into the sewer pipe, the operator aligns the camera with the centre of the pipe before starting the recording. This allows the Panoramo software to stitch the images together into a three-dimensional, virtual environment, but it also allows us to consider the images to be of the same modality, allowing a 1-on-1 comparison between two images.

As the images from the Panoramo system are meant for offline processing, the operator does not pan, rotate, or zoom the camera during the recording, as they might with other CCTV feeds. This is a very important distinction, because being able to automatically classify images where a human operator has already isolated, centred, and zoomed in on the defects, as is apparently the case in some previous studies, does not achieve much in terms of "automating classification". To take steps towards fully automated inspection, we should aim to classify images that were recorded without human intervention, other than starting the system.

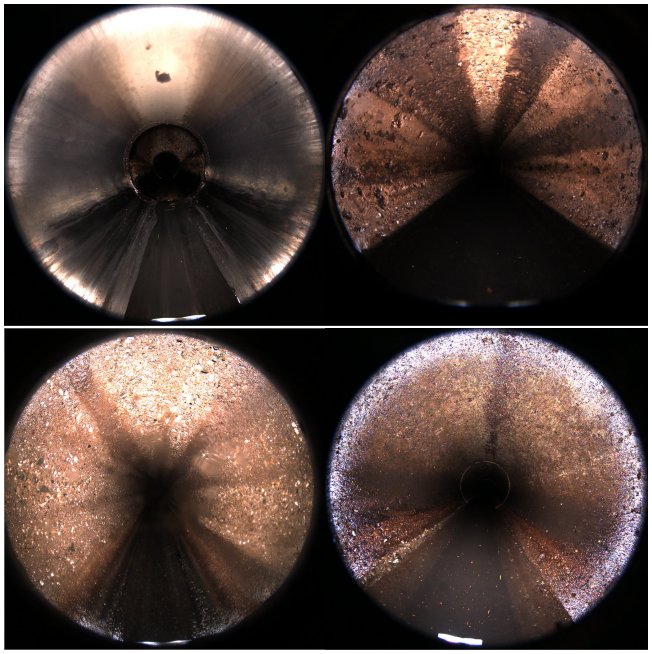


Figure 4.1: Randomly selected sample images from the dataset.

4.2.2 Inspection Reports

Besides the images, each of the 30 inspections is accompanied by an inspection report, containing all points of interest along the pipes referenced according to the European standard coding norm EN 13508-2 [17], as annotated from visual review by human operators. An example of an entry from the tabular datafile that generated this report could be

```
38.40m  BBAC2  @Blick=38.38;91;72;90;0;
```

This indicates that at 38.40 meters from the start of the pipe, a defect was found with main code `BBA` (roots), characterization `C` (complex mass of roots), and quantification `2` (pipe diameter reduced by $\leq 10\%$)

From these entries, we can assign contextual labels to the images. We have selected the twelve most commonly occurring defects (that are not simply expected landmarks

[17] European Committee for Standardization. 2003. En 13508-2: Condition of drain and sewer systems outside buildings, part 2: Visual inspection coding system, european norms

Table 4.1: Defect types and occurrences

Defect Type \ Total:	Pipes 3,350	Images 2,202,582
Fissure	586	1,442
Surface Damage	1,242	2,507
Intruding Connection	375	1,004
Defective Connection	506	838
Intruding Sealing Material	74	173
Displaced Joint	1,509	4,988
Porous Pipe	117	187
Roots	273	629
Attached Deposits	183	338
Settled Deposits	164	219
Ingress of Soil	536	1,249
Infiltration	1,353	7,565

such as pipe joints, an overview is shown in table 4.1) and matched these to specific images, using the location of the entries. Because we know the Panoramo system's images are spaced exactly 50 mm apart, it is a relatively simple task to determine which entries in the report should be visible in each image. It is important to note that the best performance we can reasonably expect to achieve on such a dataset, is to label the images as well as (and no better than) a human operator would.

The `@Blick` entry is added by the Panoramo software and can be used to recreate the exact view in the virtual environment the operator was looking at when this defect was recorded.[18] In this research, the `@Blick` entry was not used.

In the end, we have a set of roughly 2.2 million images, and for each image a list of twelve Boolean values, telling us whether or not specific defects are present in the image. Table 4.1 gives an impression of how common these defects are in the dataset. In the next section, we will go into detail on how this data is modelled.

[18] These parameters are: the location along the pipe wall, the azimuthal angle, the polar angle, the field of view angle, and the rotation of the virtual camera with respect to the water level.

4.3 Methodology

The classifier used in this research is a convolutional neural network, and the model is approximated through backpropagation. This processs is explained in more detail in section 2.3.

4.3.1 Loss Function for Multi-Label Classification

In a standard classification setting, we differentiate between different classes. Each entry in the dataset is assigned to a single class. In the case of defect detection in sewers, this leads to a problem: several defects often co-occur. Infiltration, for example, almost always has a cause that is defined as a separate defect, such as a fissure. This co-occurrence can be a result of the definitions used in the EN13508–2 guidelines [19], or it might be an effect of cascading failures [20]. In our dataset there are 17,662 out of 2,202,582 images (0.802%) that contain defects, totalling 21,139 different defects, but 6,494 of these (30.7%) co-occur with another defect in the same image. When considering entire pipes, 2,512 out of 3,350 pipes (75.0%) contain defects, 6,918 defects are found in total, and 6,171 (89.2%) of these defect types are found co-occurring with other defect types in the same pipe.

As a result of this multi-label problem,[21] we have decided to label the images with a Boolean vector, each consisting of twelve Boolean values, representing the presence or absence of a particular defect. This means that images that do not contain a defect at all will have a vector of all negatives.

Not all misclassifications should be treated equally. If we correctly classify the presence or absence of eleven defects, but misclassify the presence or absence of the last defect, this is less severe than misclassifying multiple defects.

[19] European Committee for Standardization. 2003. En 13508-2: Condition of drain and sewer systems outside buildings, part 2: Visual inspection coding system, european norms

[20] Sitzenfrei, R., Mair, M., Möderl, M., and Rauch, W. 2011. Cascade vulnerability for risk analysis of water infrastructure. *Water Science and Technology 64,* 9, 1885–1891

[21] We distinguish *multi-label* classification (multiple classes per object), as opposed to *multi-class* classification, which might also refer to a non-binary classification case, i.e. an object has a single class, but there are more than two classes.

For each of the twelve defects we calculate an individual loss function, namely the cross entropy [22] between the actual value for a defect, y_c (0 for absence, 1 for presence), and the predicted value output by the network for that defect, $\hat{y}_c$ (a real value in the interval $[0, 1]$):

[22] SHORE, J. AND JOHNSON, R. 1980. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *IEEE Transactions on information theory 26*, 1, 26–37

$$L(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{c=1}^{12} y_c \log \hat{y}_c \tag{4.1}$$

As written, only false negatives contribute to the cross entropy loss, as y_c is zero for false positives. This means that we penalise the classifier for not detecting a defect, but not for seeing a defect where there is none. To make sure that the network does not simply output 1 for all defects, $\hat{y}$ is commonly normalised so that $\sum_c \hat{y}_c = 1$, which is called *soft-max* normalization. [23] Alternatively, it is also possible to account for false positives by adding contributions both for y_c and its complement:

[23] GOODFELLOW, I., BENGIO, Y., AND COURVILLE, A. 2016. *Deep learning*. Vol. 1. MIT press Cambridge

$$L(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{c=1}^{12} y_c \log \hat{y}_c + (1 - y_c) \log(1 - \hat{y}_c) \tag{4.2}$$

This is what we will use, as normalizing $\hat{y}$ does not make much sense when we expect defects to co-occur.

4.3.2 Class Imbalance and Oversampling

Our dataset consists of 3,350 pipes with a total of 2,202,582 images. While every pipe contains at least one defect of some type in one of its images, only 17,663 images, [24] roughly 0.8% of all images contain one or more defects. It should also be noted that the percentage of pipes that contain a specific defect is not the same as the percentage of images

[24] This number is not equal to the sum of the numbers in the rightmost column of table 4.1 because defects often co-occur in the same image, and some images are counted multiple times if we sum the column.

that contain that defect, as a pipe is said to contain a defect if at least one image from that pipe contains the defect.

The extreme class imbalance of images with and without defects in our dataset means that if we train a classifier without accounting for the imbalance, it will err on the side of false negatives, as these are simply more likely. If we make some number of misclassifications, we expect these to be distributed the same as the prior probabilities of the classes, simply put: our set has about 1% defects and 99% non-defects, so of the errors, that a naive classifier will make, 1% will be a false positive and 99% will be a false negative. It can be safely assumed that a false negative is more costly than a false positive (the latter costs labour hours, the former might pose a health hazard or incur additional costs e.g. through property damage or disruption of traffic). This has some important implications for the quality assessment of a classifier as well, which have been discussed in section 2.1.4.

As noted in section 4.1.3, a lot of previous work completely disregards the class imbalance when training and testing their classifier, and instead opts for a more manageable 50% split. This approach has a major issue: the test results are not representative of a real-world scenario, and only indicative of the quality of the classifier in a general case, not for this specific classification scenario. Instead, we require the test set to have a realistic ratio of images with defects to images without defects, as this means our results translate more directly to the results we would obtain when applying our classifier on newly obtained data.

While the area-under-the-curve for an ROC-curve or a PR-curve provide a metric independent of hyperparameter selection, they still take all levels of recall into account, whereas we are likely interested only in higher levels of recall, as we have assumed that a false negative is far more costly than a false positive.

To this end, we introduce two more metrics: *specificity at recall* and *precision at recall*. Neither of these metrics

require us to manually choose a value for τ [25]. Instead they dictate τ to be chosen such that recall is at a certain level, and report the specificity (TNR) or precision at this τ. This is the same as taking a point on the ROC and PR curves that corresponds to a particular value on the recall axis, and reading the point it corresponds to on the other axis.

[25] as defined in equation (2.19)

We feel that especially the specificity at recall and precision at recall metrics are useful to put the results into real-world context: for public health reasons we might be restricted to a minimum value for recall (as a lower value would allow too many defects to slip by unnoticed and increase the risk), and we simply want to know how efficient the system is *at least* at that level. For both of these metrics, we evaluate at the recall levels $\{0.90; 0.95; 0.99\}$, as we are mostly interested in high recall. An overview of the performance metrics we are using is given in table 4.2.

4.4 Aggregating performance on pipe level

The previous section outlined performance metrics for classifying single images, but it is not an uncommon scenario to classify entire pipes as a whole for a certain defect, as the

Metric	Description
AUROC	Area under the ROC curve
AUPR	Area under the PR curve
Specificity at Recall	Percentage of non-defects detected as defects when we require a minimum percentage of defects to be detected
Precision at Recall	Percentage of detected defects that are actually non-defects when we require a minimum percentage of defects to be detected

Table 4.2: An overview of the performance metrics used

decision to intervene with repair or replacement is made on a larger scale. To achieve this, we aggregate the real and predicted labels on the images with some *aggregation rule*, and calculate the same metrics from table 4.2 on the aggregated labels.

An obvious choice for an aggregation rule is the maximum: This would be analogous to determining whether any of the images is labeled as a defect, compared to whether any of the images actually contains a defect. Importantly, this aggregation rule does not depend on the size of the pipe, like the average value would. A downside to this rule is that we might actually be detecting a defect in an image where there is none and missing a defect that is in another image, but we still count this as a true positive, because we only care to know if we found the defect in the correct pipe.

Maximum aggregation performance metrics on pipe level will be reported alongside performance metrics for single images.

4.5 Leave-two-inspections-out Cross Validation

To accurately assess performance of a classifier on a dataset, we might use k-fold cross validation [26], as outlined in Section 2.1.3. The folds are often divided either randomly or *stratified*, meaning that the classes are divided as equally as possible among the folds. Because of how our dataset was sampled, we expect a large overlap in construction material and age within an inspection, which is often performed within a single geographical neighbourhood. In this case a random or stratified split might lead to *data leakage*, information from outside the training set being implicitly part of

[26] Bishop, C. M. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg

the training set: two points in a single pipe might exhibit the same defect, as they are subject to the same conditions, are of the same build material, and have the same age. But these factors also mean that the pipes themselves might appear very similar. As a result, it might be that our classifier is simply classifying the appearance of a pipe, and not the defects themselves. If we then use random or stratified splitting, we might overestimate the actual performance.

Instead, we introduce *leave-two-inspections-out* cross validation. This is inspired by *leave-one-subject-out* cross validation, used in the medical field. Since the data is already categorised into 30 inspections, we use these same inspections as folds for cross validation. We take 28 folds as the training set, 1 fold as the testing set, and 1 fold as a validation set, to prevent overfitting on the training set. These folds are rotated 30 times, until each fold has been used as the training set once, and we have a prediction for each image. This provides a more realistic scenario, where the classifier would be used to predict the presence of defects in a pipe it has never seen before.

A possible downside of this method is that for any given fold, we might not have every defect present in both the test and validation sets. Since there is no defect that appears in fewer than three inspections, at the very least every training set will contain every defect.

4.5.1 Overfitting

Overfitting is what happens when a model is trained on the training set so well that it loses generalisability on other datasets. All data that has been sampled from real world measurements (such as photographs in our case) is expected to have some amount of noise in it. This means that any model that can describe this data to a 100% accuracy has incorporated this noise in its model. The model's performance on a different dataset (with different noise, perhaps

from different measuring instruments) will be worse than that of a model which has learnt to model the data, but not the included noise.

The risk of overfitting is exacerbated when the noise in the training set is systemic, for example through a sampling bias, as this becomes another pattern the model might detect and learn, when it is in fact noise that will not be present in future datasets. Neural networks are also more prone to overfitting than a lot of other models, because of the large number of parameters that are subject to change when learning from the training data. [27] To prevent overfitting, we employ two methods: the use of a validation set and dropout.

The use of a validation set is the more general approach of the two. Instead of training on all the data in the training set, we keep a subset of the training set apart, which is not used to train on. Periodically during the training phase, we calculate the loss function on this validation set. At some point the classifier will start overfitting, meaning the loss function on the training set will keep decreasing, but the loss function on the validation will either stagnate or start increasing. At this point we choose to stop the training and take the classifier as trained up to that point as the final classifier. We have chosen to calculate the loss on the validation set after every epoch and stop early if the loss on the validation increased significantly, or hasn't decreased for several epochs in a row.

Dropout is another way to prevent overfitting specifically for neural networks. [28] The idea is that to prevent a network that is too specifically catered to the input data, we should assure some stability with regards to small changes in the network structure. If the correct classification of a sample depends on a single specific path through the network, that classification would not be stable, as only one of the neurons in that path has to change some weights for the classification result to change. To force the network to not rely on a single path through the network, we randomly

[27] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. 2012. Improving neural networks by preventing co-adaptation of feature detectors

[28] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research 15*, 1, 1929–1958

disable neurons in the dense layers during the training step, setting their output to zero. This forces the network to create a path to the correct classification result with the still enabled neurons. Since a different set of neurons will be deactivated for every batch of data, this increases the overall stability of the network by ensuring the correct result can be reached through different paths. As the dropout is disabled after training is complete, all the paths that lead to correct classification will work together, and small changes in any one of these paths should not change the end result.

4.5.2 Averaging performance metrics across folds

While the leave-two-inspections-out cross validation should prevent data leakage and give a more accurate performance indication, it also means we are training 30 different CNNs, and combining the performance results of these into a single metric is not straightforward. There is no guarantee that the trained networks have at all similar weights at any given point or that the outputs of the networks is similar. As noted in the previous section, the distribution of defects among folds can also be skewed, with some inspections containing a lot more or fewer defects than others.

As such, it does not make sense to average the metrics as calculated on the folds. We could set a single threshold τ for each defect and fold, but since the outputs of the different networks could behave very differently, this is also not desirable.

As we have argued that it is not unlikely for defect detection systems to be tuned to achieve some minimum recall, we have decided to construct the ROC and PR curves for each fold and each defect individually, and combine the curves for different folds by equating the recall axis, and combining the values on the complimentary axis. For the

ROC curve, this is called *horizontal averaging* [29], for the PR curve, we might call it *vertical averaging*, as the recall axis is the horizontal axis, but there is no previous use of this term in literature that we know of.

It should also be noted that the averages for the specificity and precision are not calculated identically. Both are calculated with a weighted average, but the results for specificity in each fold should be weighted such that the combined result represents the specificity of the entire set, and the results for precision should be weighted such that the combined result represents the precision of the entire set. In practice, this means that the results are weighted with the relevant denominator from equation (2.16) or (2.20). As a result, a fold with no occurrences of a particular defect will have no impact on the combined specificity of that defect, and a fold with no *predicted* occurrences of a particular defect will have no impact on the combined precision of that defect.

[29] Millard, L. A. C., Kull, M., and Flach, P. A. 2014. Rate-oriented point-wise confidence bounds for roc curves. In *Machine Learning and Knowledge Discovery in Databases*, T. Calders, F. Esposito, E. Hüllermeier, and R. Meo, Eds. Springer Berlin Heidelberg, Berlin, Heidelberg, 404–421

4.5.3 Class Imbalance

Two choices have been made to adjust the classifier and make it more able to handle this imbalance: oversampling and a class-weighted loss function.

Oversampling is done from a more practical perspective: to train the CNN we have to load a *batch* of input images into memory and the backpropagation step happens for all images in the batch at once. Because of computational limitations, we found that our experimental setup could handle batches of about 50 images at a time. This means that it is extremely likely for a batch not to contain any defects at all. The gradient of such a batch can not be used to accurately estimate the gradient of the entire training set [30]. To remedy this, each image with a defect in the dataset is added not once but five times to the training set, to increase the odds of having at least one defect in every batch.

[30] Bengio, Y. 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*. Springer, 437–478

As oversampling the defects by a factor five raises the imbalance from 0.8% to about 4% of the images in the training set containing a defect, we should still be wary of training a network with such an imbalance. To shift the error that the network makes more towards false positives than false negatives, we weight the cross entropy loss function from equation (4.2) as follows:

$$L(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{c=1}^{12} W \cdot y_c \log \hat{y}_c + (1 - y_c) \log(1 - \hat{y}_c) \quad (4.3)$$

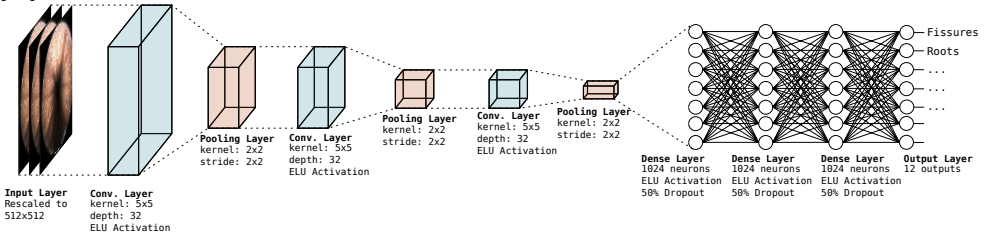
where W is a weight that represents how much more important false negatives are compared to false positives. If we consider a false negative to be 100 times more costly than a false positive, we should set W to 100.

4.6 Implementation Details

We have implemented two different CNNs, one designed by us for this task, and one reimplementation of the network used by Kumat et al. [31], which was the state-of-the-art at the time of writing (with the first layer adapted to our image sizes). This is of course not an entirely fair comparison, as we failed to reproduce their entire pipeline, but instead only replicated the network itself, but it does put the performance into context.

[31] Kumar, S. S., Abraham, D. M., Jahanshahi, M. R., Iseley, T., and Starr, J. 2018. Automated defect classification in sewer closed circuit television inspections using deep convolutional neural networks. *Automation in Construction 91*, 273–283

Figure 4.2: Network structure of our proposed CNN.

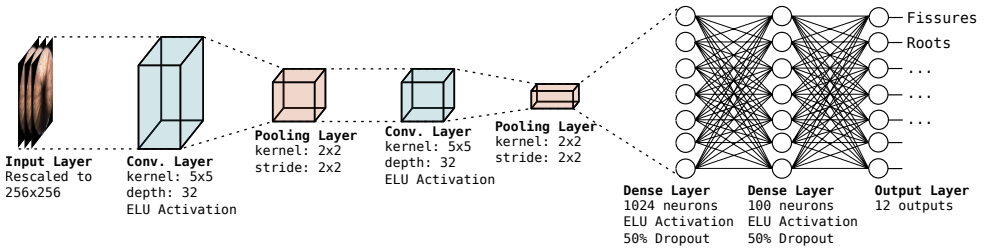


The network topologies are shown in figures 4.2 and 4.3. The network topology of our proposed CNN was designed through experimentation with different layer sizes, filter sizes, and numbers of layers on a smaller subset of the dataset.

The CNNs were built and run with TensorFlow (version 1.8.0) and Python (version 3.4.8), running on a Linux system with sixteen NVIDIA Tesla K80 GPUs and CUDA (version 9.2.148). Each network was trained using a single GPU, with several networks (one for each validation fold) being trained simultaneously on multiple GPUs. Training a single network took on average roughly five hours (per fold). Testing the different networks with each different testing fold took on average roughly 1 hour (for all 30 folds).

Each of the networks was trained with a batch size of 50 images. After every 500 batches, the performance on the validation fold was assessed. The network stopped training when the AUROC on the validation fold had not increased for 25 consecutive assessments, or when the AUROC on the validation fold decreased by more than 1%, with a minimum of 1,000 batches processed.

Figure 4.3: Network structure of the CNN proposed in Kumar et al.



4.7 Results

In this section, we present the results achieved by our proposed CNN, as well as our reimplementation of the CNN proposed by Kumar et al.,[32] on the performance metrics outlined in section 4.3.2.

Tables 4.3, 4.4, 4.5, and 4.6 show the specificity (TNR) and precision at recall (TPR) values of 0.90, 0.95, and 0.99, for our proposed CNN and our reimplementation of the CNN proposed by Kumar et al. The better result for each scenario is displayed in bold when it is significantly better, determined by a paired sample t-test (at a significance level of $\alpha = 0.05$) across the validation folds.

Figures 4.8.1, 4.8.1, 4.8.1, and 4.8.1 show the ROC and PR curves for our proposed CNN for classification in images and entire pipes.

[32] Kumar, S. S., Abraham, D. M., Jahanshahi, M. R., Iseley, T., and Starr, J. 2018. Automated defect classification in sewer closed circuit television inspections using deep convolutional neural networks. *Automation in Construction 91*, 273–283

4.8 Discussion

Looking at tables 4.3, 4.4, 4.5, and 4.6, we see that in each of the shown scenarios, our proposed CNN either outperforms Kumar et al.,[33] or it does not perform significantly worse. Out of 144 scenarios, our proposed network wins significantly 81 times. Additionally, it wins 44 times, but not by a significant margin, and looses 19 times, but never significantly.

[33] Kumar, S. S., Abraham, D. M., Jahanshahi, M. R., Iseley, T., and Starr, J. 2018. Automated defect classification in sewer closed circuit television inspections using deep convolutional neural networks. *Automation in Construction 91*, 273–283

4.8.1 Classifying individual images

When we take a closer look at the ROC and PR curves for the classification of individual images in figures 4.8.1 and 4.8.1, there are a few observations to be made.

The ROC curves in figure 4.8.1 generally look quite good, with the exception of those for porous pipes, and to a lesser

degree attached deposits and settled deposits. The class imbalance is quite important here: the AUROC does not take into account that a false positive and false negative are not comparable in this context. As noted earlier, we are mostly interested in the scenario with high recall (TPR), as these are a requirement for any kind of automated sewer inspection, which means the top portion of each plot is more important than the bottom portion. It must also be noted that while both axes go from 0 to 1, the horizontal axis represents many more images than the vertical axis does, because of the class imbalance. One interesting feature of these curves, is that they seem to have a 'plateau' near the top. This indicates that a specific threshold exists where it is no longer advantageous to further increase the threshold, as this will only increase the false positive rate, but not the true positive rate. The false positive rate at this interval (which is approximately equal to 1 minus the specificity at 99% recall, as noted in table 4.3) can be regarded as the best specificity we can achieve for a certain defect.

The PR curves in figure 4.8.1 paint a different picture: the PR curves are mostly below an F_1-score of 0.2, seeming very unimpressive. Similar to the ROC curves, we are mostly interested in high recall, i.e. the rightmost portion of each plot. In this case, the precision seems to be quite low, but unlike the specificity, the precision axis *is* scaled with the prior probability of the defects. We will go into more detail on how to interpret these precision scores in the next section, but it should be noted that a small precision is expected when we have small prior probabilities.

4.8.2 Classifying entire pipes

When we observe the ROC and PR curves for classification of pipes in figures 4.8.1 and 4.8.1, they paint a rather different image. The ROC curves in figure 4.8.1 do not look very good, but it should be noted that the 'plateaus' are

Table 4.3: **Specificity at recall** values when classifying **single images**. Numbers displayed in bold indicate that performance is significantly better than the performance achieved by the other network, as determined by a paired sample t-test at significance level $\alpha = 0.05$.

	Specificity at 0.90 recall		Specificity at 0.95 recall		Specificity at 0.99 recall	
Defect	This work	Kumar et al.	This work	Kumar et al.	This work	Kumar et al.
Fissure	**0.754**	0.375	**0.683**	0.290	**0.550**	0.208
Surface Damage	**0.702**	0.240	**0.548**	0.107	**0.291**	0.045
Intruding Connection	**0.916**	0.448	**0.809**	0.392	**0.741**	0.370
Defective Connection	**0.901**	0.553	**0.811**	0.460	**0.703**	0.372
Intruding Sealing Material	**0.780**	0.061	**0.731**	0.057	**0.706**	0.057
Displaced Joint	**0.691**	0.441	**0.532**	0.306	**0.262**	0.145
Porous Pipe	0.349	0.207	**0.322**	0.179	0.307	0.171
Roots	**0.728**	0.209	**0.633**	0.166	**0.561**	0.159
Attached Deposits	**0.388**	0.142	**0.313**	0.116	**0.281**	0.115
Settled Deposits	**0.510**	0.114	**0.459**	0.102	**0.442**	0.097
Ingress of Soil	**0.762**	0.322	**0.670**	0.237	**0.532**	0.180
Infiltration	**0.622**	0.220	**0.486**	0.160	**0.253**	0.092

Defect	Precision at 0.90 Recall		Precision at 0.95 Recall		Precision at 0.99 Recall	
	This work	Kumar et al.	This work	Kumar et al.	This work	Kumar et al.
Fissure	**0.036**	0.006	**0.019**	0.004	**0.005**	0.003
Surface Damage	**0.011**	0.003	**0.005**	0.002	**0.003**	0.002
Intruding Connection	**0.071**	0.007	**0.011**	0.005	**0.006**	0.004
Defective Connection	**0.014**	0.002	**0.008**	0.002	**0.004**	0.001
Intruding Sealing Material	0.002	0.000	0.002	0.000	0.001	0.000
Displaced Joint	**0.015**	0.006	**0.010**	0.005	**0.005**	0.004
Porous Pipe	0.000	0.000	0.000	0.000	0.000	0.000
Roots	**0.003**	0.001	**0.002**	0.001	**0.002**	0.001
Attached Deposits	**0.001**	0.000	**0.001**	0.000	**0.001**	0.001
Settled Deposits	0.001	0.000	0.000	0.000	0.000	0.000
Ingress of Soil	**0.008**	0.002	**0.005**	0.002	**0.003**	0.002
Infiltration	**0.024**	0.012	**0.017**	0.012	**0.013**	0.012

Table 4.4: **Precision at recall** values when classifying **single images**. Numbers displayed in bold indicate that performance is significantly better than the performance achieved by the other network, as determined by a paired sample t-test at significance level $\alpha = 0.05$.

Table 4.5: **Specificity at recall** values when classifying **entire pipes**. Numbers displayed in bold indicate that performance is significantly better than the performance achieved by the other network, as determined by a paired sample t-test at significance level $\alpha = 0.05$.

	Specificity at 0.90 recall		Specificity at 0.95 recall		Specificity at 0.99 recall	
Defect	This work	Kumar et al.	This work	Kumar et al.	This work	Kumar et al.
Fissure	**0.428**	0.357	**0.365**	0.279	0.306	0.261
Surface Damage	0.250	0.226	0.193	0.155	0.128	0.107
Intruding Connection	**0.414**	0.250	**0.371**	0.224	**0.354**	0.220
Defective Connection	**0.230**	0.186	0.186	0.147	0.172	0.132
Intruding Sealing Material	0.411	0.406	0.411	0.406	0.411	0.406
Displaced Joint	0.294	0.268	**0.219**	0.169	**0.123**	0.085
Porous Pipe	0.363	0.399	0.346	0.377	0.344	0.372
Roots	0.403	0.338	0.338	0.290	0.317	0.267
Attached Deposits	0.518	0.481	0.496	0.454	0.486	0.444
Settled Deposits	**0.386**	0.305	**0.364**	0.282	**0.363**	0.282
Ingress of Soil	0.285	0.315	0.237	0.286	0.209	0.245
Infiltration	0.284	0.298	0.218	0.207	0.141	0.133

Defect	Precision at 0.90 Recall		Precision at 0.95 Recall		Precision at 0.99 Recall	
	This work	Kumar et al.	This work	Kumar et al.	This work	Kumar et al.
Fissure	**0.414**	0.379	**0.393**	0.364	0.363	0.358
Surface Damage	0.582	0.589	0.573	0.570	0.557	0.552
Intruding Connection	**0.369**	0.333	**0.360**	0.324	**0.348**	0.315
Defective Connection	**0.294**	0.276	**0.291**	0.273	**0.285**	0.272
Intruding Sealing Material	0.068	0.062	0.068	0.062	0.068	0.062
Displaced Joint	0.600	0.602	0.585	0.582	0.565	0.561
Porous Pipe	0.130	0.137	0.129	0.136	0.129	0.135
Roots	**0.208**	0.189	0.185	0.184	0.176	0.175
Attached Deposits	0.190	0.196	0.185	0.182	0.183	0.179
Settled Deposits	0.125	0.118	0.117	0.108	0.117	0.109
Ingress of Soil	0.363	0.364	0.348	0.358	0.339	0.335
Infiltration	0.584	0.602	0.579	0.584	0.560	0.563

Table 4.6: **Precision at recall** values when classifying **entire pipes**. Numbers displayed in bold indicate that performance is significantly better than the performance achieved by the other network, as determined by a paired sample t-test at significance level $\alpha = 0.05$.

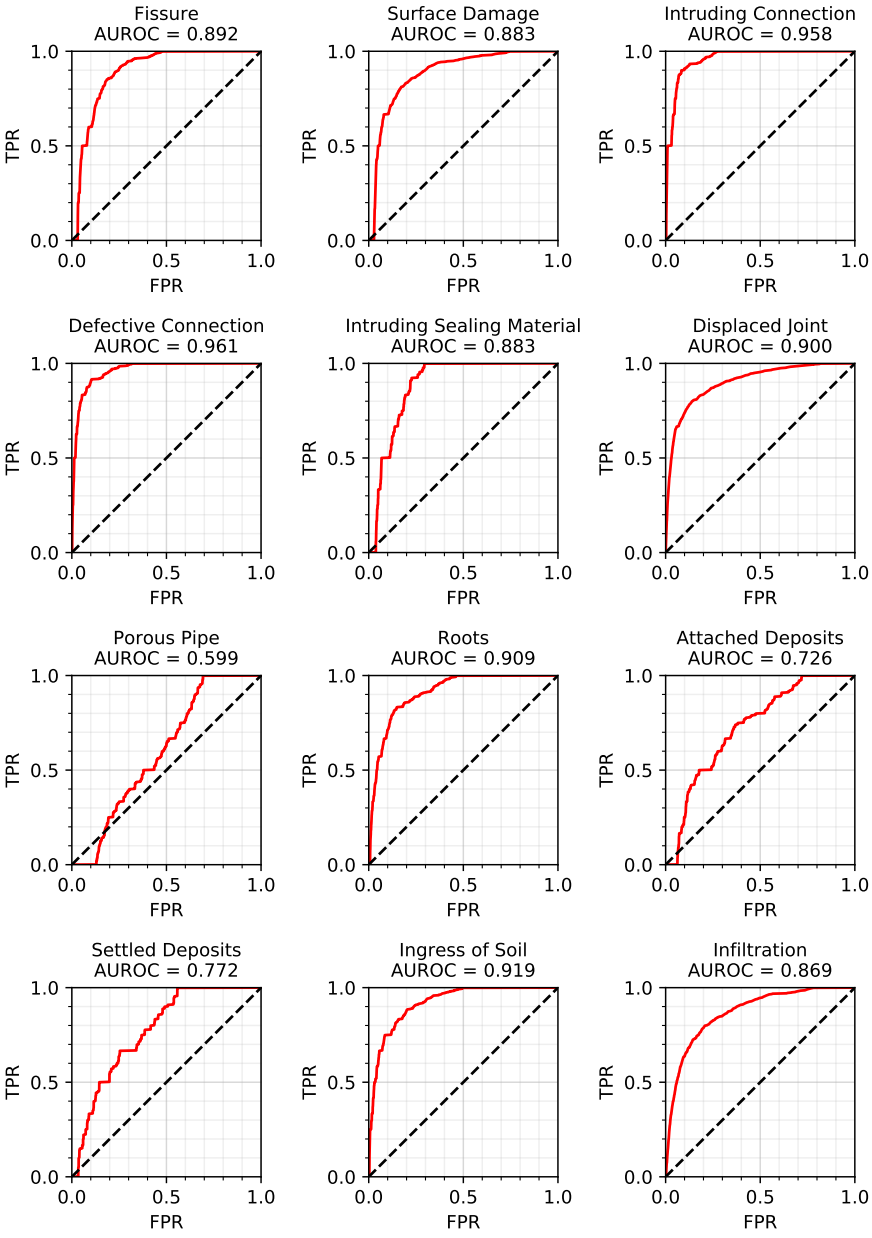


Figure 4.4: **ROC Curves** for the proposed CNN when classifying **single images**.

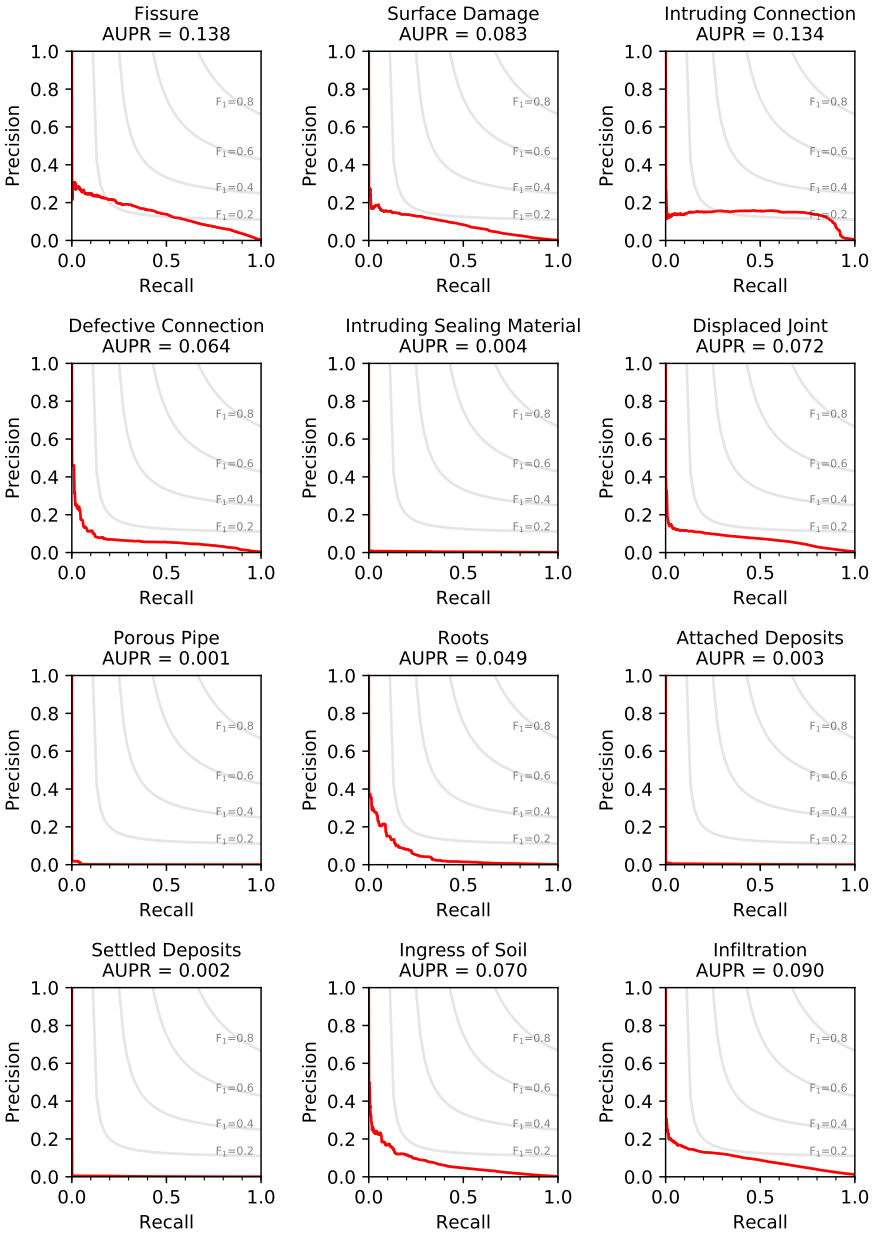


Figure 4.5: **Precision-Recall Curves** for the proposed CNN when classifying **single images**.

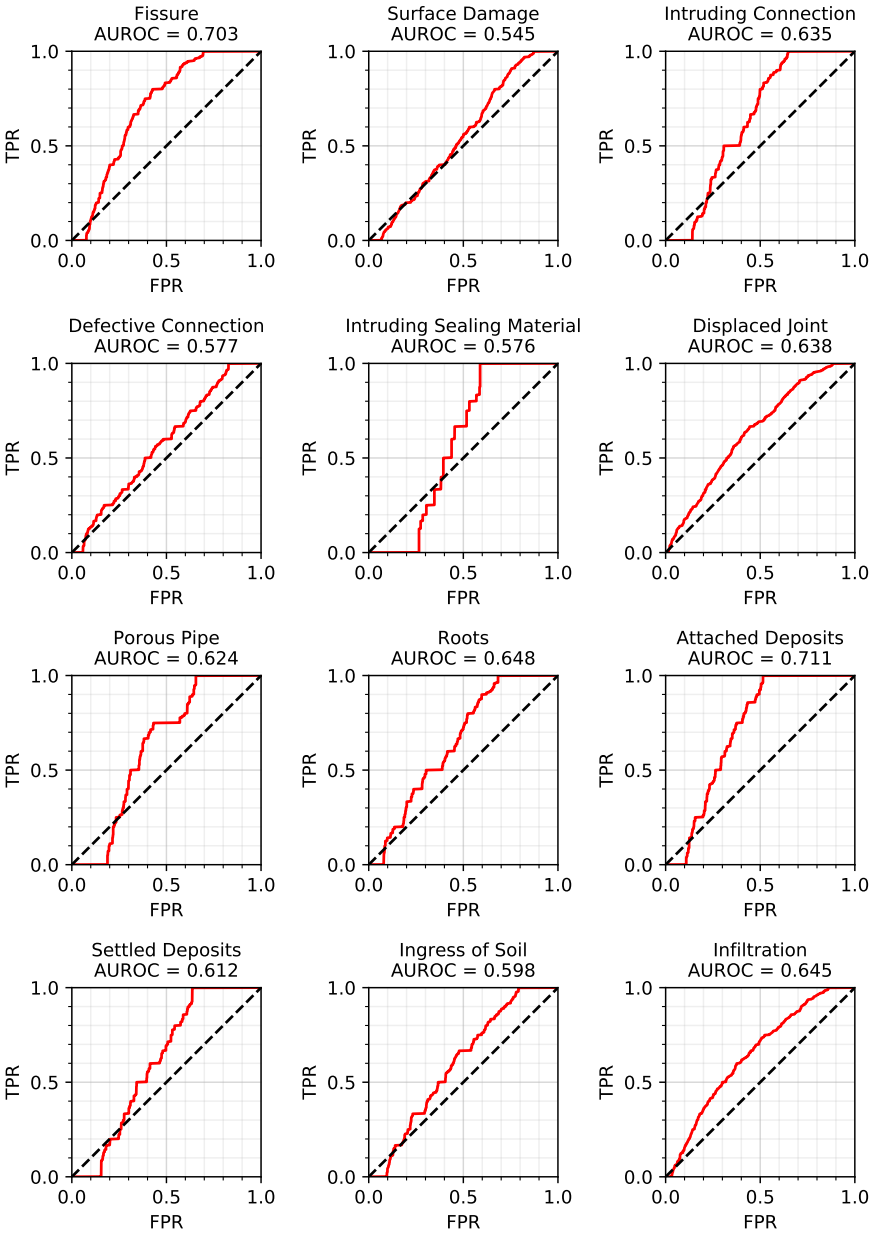


Figure 4.6: **ROC Curves** for the proposed CNN when classifying **entire pipes**.

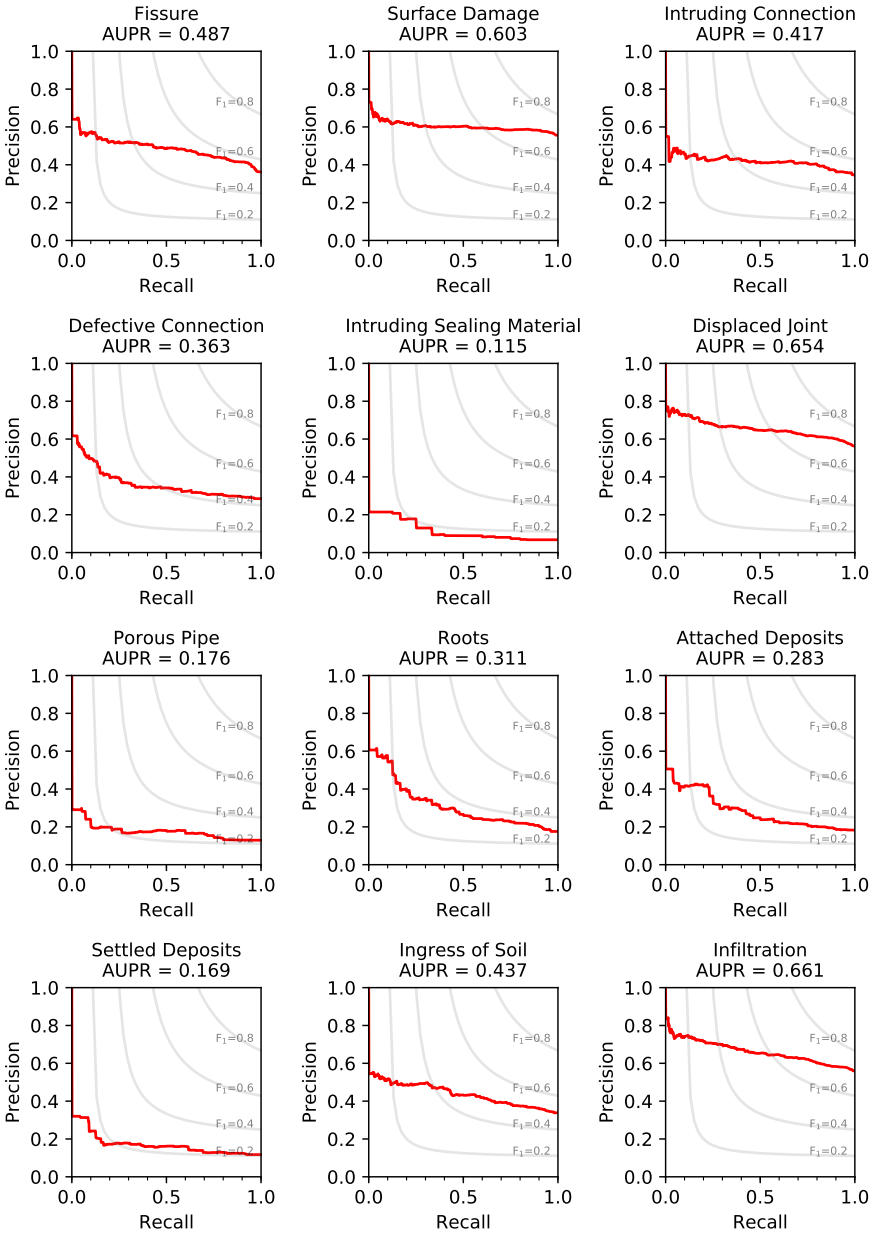


Figure 4.7: **Precision-Recall Curves** for the proposed CNN when classifying **entire pipes**.

again present in a lot of the curves, which again indicates that there are some pipes of which we are confidently sure they *do not* contain defects. Rather interestingly, among the better ROC curves are those that underperformed on single-image classification: porous pipes, attached deposits, and settled deposits. This might indicate that some labels were missing in our dataset: if a pipe has these defects at multiple locations but only a few locations were marked in the inspection report, we would overestimate the false positives our classifier finds in single-image classification, but we can be more sure when deciding whether a pipe has or does not have this defect.

The PR curves for the classification of pipes in figure 4.8.1 looks a lot better than that of single images. This is because the class imbalance is much less present on pipe-level. Still the worst results are obtained for classes that have a low prior on pipe-level (intruding sealing material, porous pipe, roots, attached deposits, settled deposits), as expected for the precision.

4.8.3 Result Interpretation

To put our findings into context, we will take a closer look at their impact on the day-to-day operation of sewer inspections aided by our automated system. When looking at our results superficially, they are easily misinterpreted. It is important to keep in mind that we are dealing with a very imbalanced dataset, which makes the precision the more interesting of these metrics (as described in section 4.3.2). Let's consider the class *Fissure* in more detail. From table 4.1 we can tell that approximately 0.065% of the images (1,442 out of 2,202,582 images in total) contain a fissure, which makes for a very imbalanced target. For fissures at 0.90 recall we achieve a specificity of 0.754 and a precision of 0.036 (see tables 4.3 and 4.4, top left cell). [34]

[34] It should be noted that these numbers do not add up perfectly to the 1,442 fissures out of 2,202,582 images, as the specificity and precision are aggregated over 30 different folds, each with its own specificity and precision.

The specificity of 75.4% indicates that, of all the images that do *not* contain fissures, we identify 75.4% as such, and the remaining 24.6% are suspected of containing fissures, meaning they still have to be inspected by an operator. The precision indicates that of all fissure detections, 3.6% are true positives, while the remaining 96.4% are false alarms.

If we assume that fissures are randomly distributed, an unsupported operator would have to inspect 90% of all images ($0.9 \times 2{,}202{,}582 = 1{,}982{,}324$ images) to find 90% of the fissures. Our proposed classifier detects 90% of all images with fissures with a specificity of 75.4%.

To detect 90% of all fissures, an operator would have to inspect all detections the system made: $0.246 \times (2{,}202{,}582 - 1{,}442) + 0.9 \times 1{,}442 = 542{,}778$ images. In comparison to the situation without a classification system, this is equal to a reduction of 72.6%. In an ideal situation, this means that the time an operator spends on inspecting fissures is reduced by almost a factor 4. Table 4.7 lists these reduction numbers (derived from tables 4.3 and 4.4) for all defect types considered. The reduction of 72.6% for Fissure appears as the top-left cell. The highest reduction (at 0.90 recall) is attained for Intruding Connection, with a 90.7% reduction (a factor 10). Not surprisingly, this defect type scores well both in the ROC as in the PR plots. It ranks 6th in terms of frequency of defect type, with 1,004 observed cases.

We can perform the same calculations with the results from classification on *entire pipes*, but the interpretation is a little less clear, as we cannot assume different pipes take the same amount of time for review; especially pipes with a lot of defects will be more labour-intensive to inspect. From table 4.1 we can tell that approximately 17.5% of pipes contain fissures (586 out of 3,350 pipes). Let us for this case assume 99% of all pipes containing fissures need to be detected, this means $0.99 \times 3{,}350 = 3{,}317$ pipes have to be inspected for fissures. Our classifier achieves 99% recall with a specificity of 30.6% (table 4.5) and a precision of 36.6% (table 4.6). By

Table 4.7: Reduction of *images* that need to be reviewed by a human after inspection with our classifier, expressed in percentage compared to images that would need to be inspected without our classifier.

	Recall		
Defect Type	**0.90**	**0.95**	**0.99**
Fissure	72.6%	66.6%	54.5%
Surface Damage	66.8%	52.4%	28.3%
Intruding Connection	90.7%	79.8%	73.8%
Defective Connection	89.0%	80.1%	70.0%
Intruding Sealing Material	75.5%	71.7%	70.3%
Displaced Joint	65.5%	50.7%	25.4%
Porous Pipe	27.7%	28.7%	30.0%
Roots	69.8%	61.4%	55.6%
Attached Deposits	32.0%	27.7%	27.4%
Settled Deposits	45.5%	43.1%	43.7%
Ingress of Soil	73.5%	65.3%	52.7%
Infiltration	57.8%	45.8%	24.4%

the same calculations as before, this means we now have to inspect $0.694 \times (3{,}350 - 586) + 0.99 \times 586 = 2499$ pipes. This is a reduction of 24.7%. Table 4.8 shows similar reductions for all the defect types, for pipes.

In table 4.7 we see that intruding and defective connections are best classified by our CNN and have the largest reduction rate in images or pipes that still require human review, while porous pipes are the more difficult to classify and these have the lowest reduction rates.

Realistically, the defects are not randomly distributed throughout the image set and operators would not inspect single images, but rather a sequence of images with a clear spatial relationship (a 5 cm shift). This means that the reduction by a factor of 4 is almost certainly an overestimation. On the other hand, we know defects can often co-occur [35] and this estimation was only for fissures, which has one of the higher prior probabilities of the defects we consider. For defects with a lower prior probability, there is a larger potential for improvement.

It should also be noted that with the reported false negative probability of about 25%[36] in the labels of our data set,

[35] SITZENFREI, R., MAIR, M., MÖDERL, M., AND RAUCH, W. 2011. Cascade vulnerability for risk analysis of water infrastructure. *Water Science and Technology 64,* 9, 1885–1891

[36] DIRKSEN, J., CLEMENS, F., KORVING, H., CHERQUI, F., LE GAUFFRE, P., ERTL, T., PLIHAL, H., MÜLLER, K., AND SNATERSE, C. 2013. The consistency of visual sewer inspection data. *Structure and Infrastructure Engineering 9,* 3, 214–228

Defect Type	Recall 0.90	0.95	0.99
Fissure	30.1%	27.4%	24.7%
Surface Damage	10.5%	9.5%	7.5%
Intruding Connection	31.0%	30.0%	30.9%
Defective Connection	12.2%	12.1%	13.9%
Intruding Sealing Material	33.8%	37.2%	39.7%
Displaced Joint	11.9%	9.8%	6.3%
Porous Pipe	28.3%	30.1%	32.6%
Roots	30.9%	27.8%	28.5%
Attached Deposits	43.9%	44.3%	45.4%
Settled Deposits	30.3%	31.5%	33.9%
Ingress of Soil	17.2%	16.5%	16.9%
Infiltration	12.2%	10.5%	7.9%

Table 4.8: Reduction of *pipes* that need to be reviewed by a human after inspection with our classifier, expressed in percentage compared to pipes that would need to be inspected without our classifier.

the actual precision and specificity are likely higher than we report. For any given defect, there is approximately a 1 in 4 chance that the operator missed it and it was labeled in our dataset as not being a defect (whereas the probability of a false positive was estimated "in the order of a few percent"). The 1,442 images that are labeled as fissures, are possibly only 75% of all images labeled containing fissures, meaning there would be approximately 480 images among the images not labeled as fissures.

4.8.4 Combining Defect Outputs

Because of the co-occurrence of defects, it can be interesting to combine the classifier outputs for different defects into a single decision: *"Does this image/pipe need further (human) review?"*

As discussed in section 4.3, in our dataset 30.7% of defects in images co-occur with other defects in the same image and 89.2% of defects in pipes co-occur with other defects in the same pipe. To treat the problem as a binary classification problem, we simply take the maximum value of the true

Table 4.9: Specificity and precision at recall values for binary classification on either single images or entire pipes.

Metric and Classification type	Recall 0.90	0.95	0.99
Specificity for Images	0.649	0.452	0.180
Specificity for Pipes	0.372	0.284	0.113
Precision for Images	0.021	0.014	0.009
Precision for Pipes	0.717	0.703	0.668

label over the classes (a 1 if at least one defect is present, a 0 if no defects are present), and the average of the predicted labels over the classes (a real-valued number between 0 and 1). This gives us the curves as shown in figure 4.8.

For classification on images, reducing this problem to a binary classification case does not improve things much. The overall result is approximately equal to the average of the classification results on individual classes. This is not unexpected, as the co-occurrence of defects in individual images is rare.

For classification on pipe level though, the results are

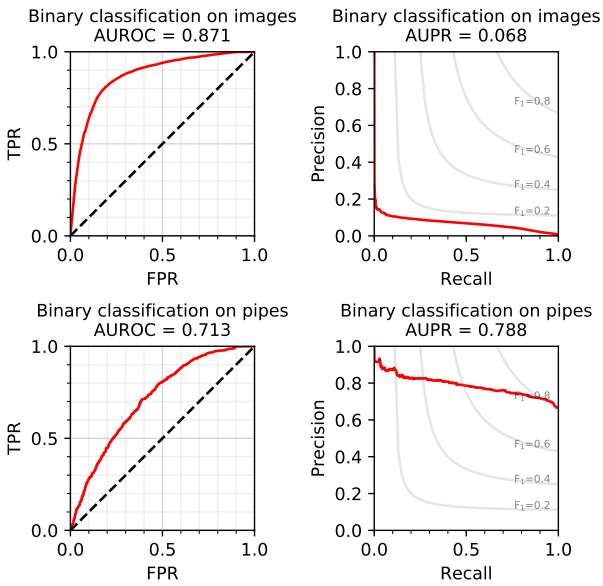


Figure 4.8: ROC and PR curves obtained when treating the problem as a binary classification problem, for image level or pipe level.

Classification Type	Recall 0.90	0.95	0.99
Images	60.5%	42.0%	17.0%
Pipes	7.6%	6.2%	2.6%

Table 4.10: Reduction of images or pipes that need to be inspected with our combined binary classifier, expressed in percentage compared to pipes that would need to be inspected without our combined binary classifier.

more interesting than a simple averaging. The PR curve is strictly better than the PR curves of individual defects. The ROC curve at high recall is slightly worse than some individual defects, but the overall AUROC is higher.

Table 4.9 shows the specificity and precision at specific recall values, for comparison with the multi-label classification results in tables 4.3, 4.4, 4.5, and 4.6. Using these values we can again calculate the reduction in images or pipes that require review to achieve a certain recall, as shown in table 4.10.

The reductions on pipe level are quite low, this is because the transition to a binary classification scenario results in the class imbalance disappearing on pipe-level: 75.0% of pipes contain *at least one* defect, and fall into the positive class. This means a high precision is required, and while precision had increased by combining the defect types, the reduction has decreased.

4.9 Conclusion

In this chapter, we have approached the task of automated defect detection in sewer image sets as a supervised classification task. The focus has been on the validation methodology used to interpret the results achieved by a classifier. While we feel that there is a lot of potential for future improvement of classifiers trained for this task, with the data and computational resources available, the proposed convolutional neural network performed reasonably well.

While our proposed classifier does not perform well enough for fully autonomous classification, it can be used to signif-

icantly reduce the amount of images that require human review by eliminating images which are highly unlikely to contain defects according to the classifier. We estimate the amount of images that require human review can be reduced by 60.5%, given that detecting 90% of all defects is sufficient.

We compared the results of our proposed classifier to that of Kumar et al.,[37] and our proposed classifier outperforms their proposed classifier, but we did not implement their classification pipeline beyond the network structure, such as for example, the oversampling outlined in their work. Our dataset also differs significantly from theirs. As noted in section 4.2, no human inspector has changed the camera settings during the inspection, as is common with other CCTV inspection datasets.

[37] Kumar, S. S., Abraham, D. M., Jahanshahi, M. R., Iseley, T., and Starr, J. 2018. Automated defect classification in sewer closed circuit television inspections using deep convolutional neural networks. *Automation in Construction 91*, 273–283

A major topic of this chapter was the validation methodology. We have discussed our reasons for choosing the "specificity at recall" and "precision at recall" metrics for this specific task in section 4.3.2: these give us easily interpretable measures of the possible improved efficiency at realistic scenarios. We have also explained why "leave-two-inspections-out cross validation" is an appropriate way to prevent data leakage, and applied this technique in our experiments. These methods provide us with less biased and more easily interpretable results.

4.9.1 Future Work

Not all information in the inspection reports was used to its full potential and we feel that using the information pertaining to where in an image a defect is visible (with a classifier capable of processing this information of course) could lead to further performance improvement. Additionally, the use of other types of sensors, either already present on or easily added to the pipe inspection vehicle, may prove to be useful for further improvement.

Since we know there are likely undetected defects in our dataset,[38] it would be an interesting experiment to see if a classifier trained on data where these are unlabeled, is still able to find these defects in its own training set. To achieve this, the false positive detections would have to be re-classified by a human operator. Hopefully, this would indicate that the classifier detected defects that we *thought* were false positives, but were in fact true positives. Unfortunately, this is beyond the scope of this thesis.

[38] Dirksen, J., Clemens, F., Korving, H., Cherqui, F., Le Gauffre, P., Ertl, T., Plihal, H., Müller, K., and Snaterse, C. 2013. The consistency of visual sewer inspection data. *Structure and Infrastructure Engineering 9,* 3, 214–228