Universiteit Leiden
The Netherlands

Opleiding Informatica

Using Convolutional Autoencoders

to Detect Anomalies in Sewer Images

Mitchell Kesteloo

Supervisors:
Dirk Meijer & Arno Knobbe

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

20/07/2017

Abstract

The Dutch sewer system is currently inspected manually, which leads to inconsistent and subjective inspections. This is why the SewerSense project was started, which aims to automate the inspection process. In this thesis, a convolutional autoencoder (CAE) was designed to see if this method is a viable option for the use of anomaly detection. First, the proof of concept is illustrated using the MINST and CIFAR-10 image sets. The CAE proved to perform excellent on these sets. An experiment was conducted to analyse whether the CAE also performs well on the sewer pipe image sets. Two image sets were available: one with smooth concrete and one with coarse agglomerate. The CAE has an acceptable performance on the smooth concrete image set, but performs unsatisfiable on the coarse agglomerate image set.

Contents

1 Introduction **1**

1.1 Background . . . 1

1.2 Research objectives . . . 1

1.3 Thesis Overview . . . 3

2 Preliminaries **4**

2.1 Convolutional Neural Network . . . 4

2.1.1 Convolution . . . 4

2.1.2 Pooling . . . 6

2.2 Autoencoder . . . 7

2.3 Convolutional Autoencoder . . . 8

2.3.1 Unpooling . . . 8

2.3.2 Deconvolution . . . 9

2.3.3 Overall Structure . . . 10

3 Methods **11**

3.1 TensorFlow . . . 11

3.2 MNIST . . . 11

3.3 CIFAR-10 . . . 12

3.4 Network Structure . . . 12

3.5 Training and Validating the Network . . . 13

3.5.1 Training . . . 13

3.5.2 Validation . . . 13

4 Experiment **17**

4.1 Sewerpipe Image Sets . . . 17

4.1.1 Available data . . . 18

4.2 Network Structure . . . 19

4.3 Validation . . . 20

5 Results **21**

5.1 Smooth Image Set 21
5.2 Coarse Image Set 23
5.3 Comparing Smooth and Coarse 25

6 Conclusion **29**
6.1 Summary 29
6.2 Future Work 30

Bibliography **32**

A Scientific Research Paper **34**

Chapter 1

Introduction

This thesis is about the SewerSense project which is led by researchers from Leiden University and TU Delft. Some background information is given in Section 1.1. The research objectives will be defined in Section 1.2 and an overview of this thesis will be given in Section 1.3.

1.1 Background

The Dutch sewer system consists of around 100,000 km of sewer pipes. Because the sewer system is crucial for public health, it is essential that one can detect defects as early as possible. When a defect has been detected, a decision has to be made as to what kind of actions are needed to repair these defects. Currently, the sewers are checked manually. A technician lowers a camera in the sewer system that takes pictures every couple of centimeters. The technician then checks this material for any defects. These actions are performed roughly once every ten years for every stretch of pipe.
One can imagine this process is costly and time consuming. Furthermore, the quality of the video material is not perfect: technicians tend to miss some defects and it is difficult for policymakers to determine what actions are needed because information is lost during the processing of the data.
Following the above, one can say that there is room for improvement. SewerSense [1] is a project where researchers will try to create an automated system which will detect defects in the sewer system.

1.2 Research objectives

The goal of this research is to investigate whether using a convolutional autoencoder (CAE) is a viable option to detect anomalies in image sets. The problem is that a lot of research done with CAEs is about classifying objects, so there is not a large amount of related work. This project is about detecting small (sometimes minute)

defects. Data might get lost during pooling, so when running the program, this lost data might just contain information about a tiny defect. Lets consider Figure 1.1. If there is a small defect, say a root that just barely grew through the concrete, it will be hard to notice that. However, the image sets contain a huge amount of images of sewer pipes without defects. This means that even the smallest of defects are likely to be detected after training the network. The output of the trained CAE will not show any sign of the minor defect, but this means that the cost will show that there is a difference.

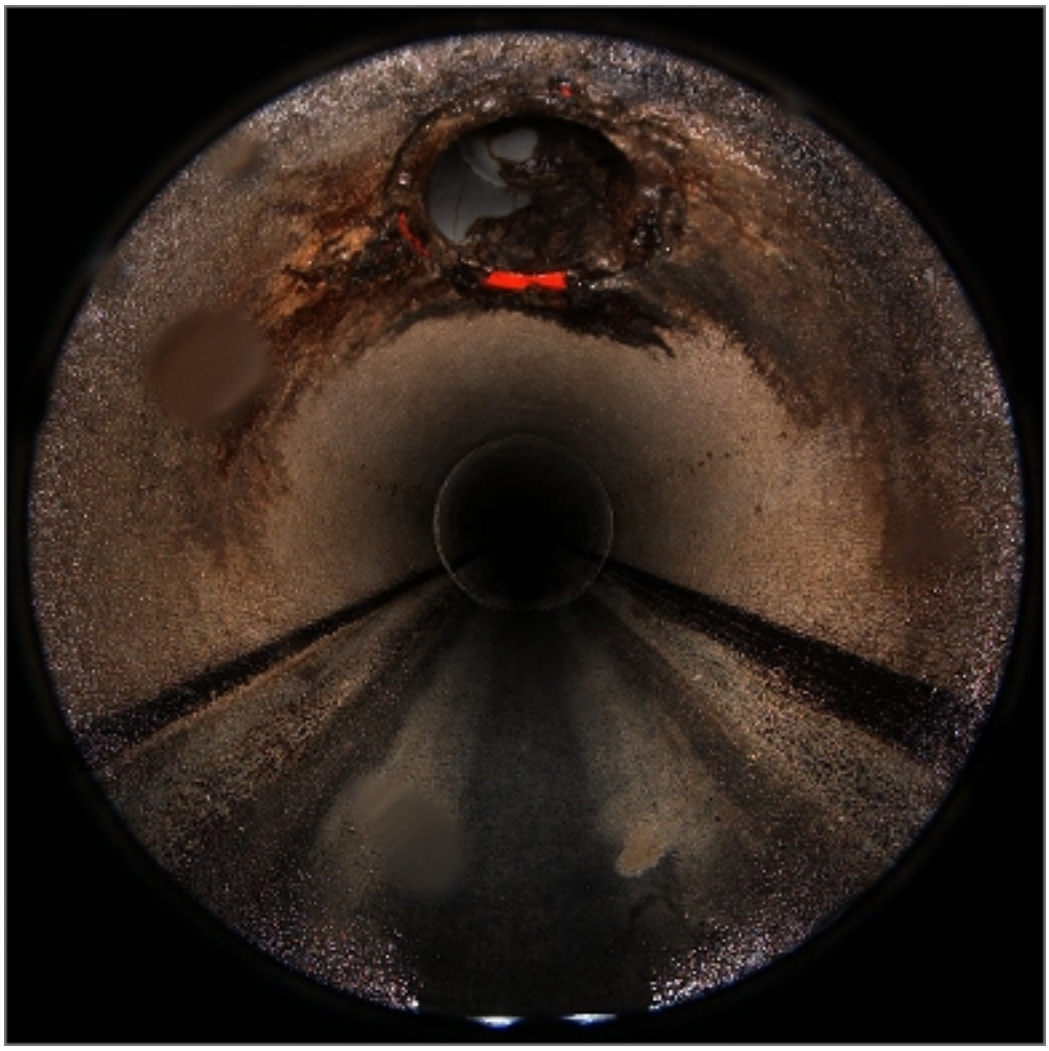


Figure 1.1: Picture of a sewer used in the project [1]

Based on all given information, the following research question is posed:
Is the use of convolutional autoencoders viable for detecting anomalies in aligned image sets?
For this problem, there is but one specific requirement for the image set: **all images have to be aligned.** If the image set does not meet this requirement, the CAE will not be able to train accurately. The image sets used for this project meets this requirement.

1.3 Thesis Overview

Chapter 2 contains some background information about convolutional neural networks and autoencoders. Chapter 3 explains what software and methods are used in this project. In Chapter 4, the experiment with the sewer pipe image sets is described. The results of this experiment are presented in Chapter 5. Finally, the conclusions and recommendations for future research are presented in Chapter 6.

This thesis is part of the bachelor program of Computer Science at the Leiden Institute of Advanced Computer Science (LIACS) at Leiden University. This thesis was written under the supervision of Dirk Meijer and Arno Knobbe.

Chapter 2

Preliminaries

2.1 Convolutional Neural Network

A convolutional neural network (CNN) carries out two operations beyond a regular neural network: convolution and pooling [2]. The input can go through several layers of convolution and pooling. The output after convolution and pooling is usually the input for a "regular" neural network. See Figure 2.1 for a CNN with one convolutional and pooling layer.

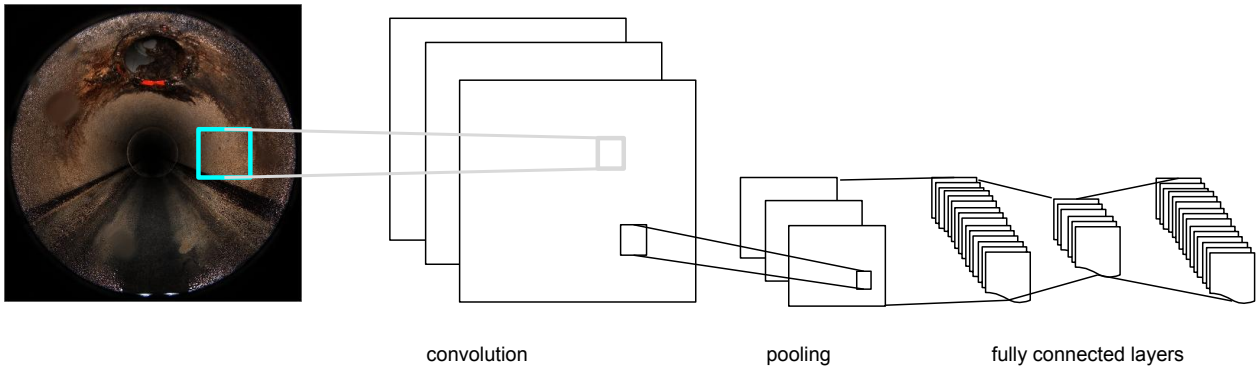


Figure 2.1: A basic CNN

2.1.1 Convolution

The first step, convolution, is equivalent to sliding a **filter** over an input matrix. The convolutional layer consists of a set of these trainable filters. The filter is convolved across the width and height of the input volume and the dot products between the entries of the filter and the input is computed at every position. Convolution can be mathematically described by the following equation [3]:

$$w(x,y) * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t) \quad (1)$$

Here, w represents a filter of size m by n and f represents the image. x and y are the displacement variables. For a filter of size m by n, the image is zeropadded with at least m - 1 rows at the top and bottom and the image is zeropadded with at least n - 1 rows at the right and left. a is then equal to $\frac{m-1}{2}$, b is equal to $\frac{n-1}{2}$ and for notational convenience, m and n are assumed to be odd integers. This equation is evaluated for every value of x and y to make sure that every element of w visits every pixel in f. An example can be seen in Figure 2.2.

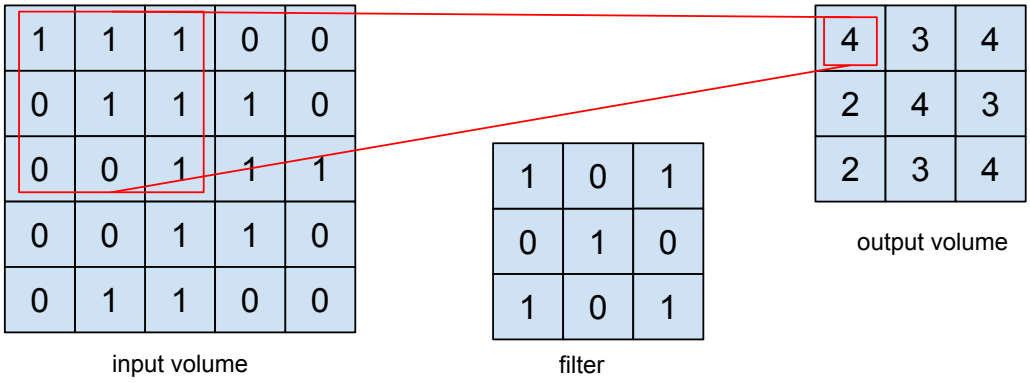


Figure 2.2: Convolution with a 3x3 filter

The output volume depends on the **depth** and the **stride**. The depth is the amount of filters used. A bigger depth means more extracted features. The stride is the amount of pixels you move the filter at one time which, when looking at equation (1), means that instead of incrementing the displacement variables x and y by one, these are incremented by the wanted strides. The smaller the stride, the bigger the output volume.

There are different ways to handle convolution around the edges of an image. One of them is zero padding, which, as the word suggests, means adding zeroes around the images. Another way is taking the pixel value of an edge and use this value for padding. In this project, zero padding is used, because it will be easier to design the network (no issues with reshaping because of differences in dimensions when going from one layer to the other).

After convolving, non-linearity is introduced. This is done because the physical model (lighting, texture, positioning) which produced these images is non-linear, so the assumption is that with a non-linear model, the images will be constructed more accurately. An activation function is used to do this. Some well known activation functions are the sigmoid function and the hyperbolic tangent function. In this case, the **ReLU** (rectified linear unit) is used. It appears that in practice this function is a good choice for CNNs [4]. The ReLU function replaces all negative values with zeroes: $f(x) = \max(0, x)$, see Figure 2.3.

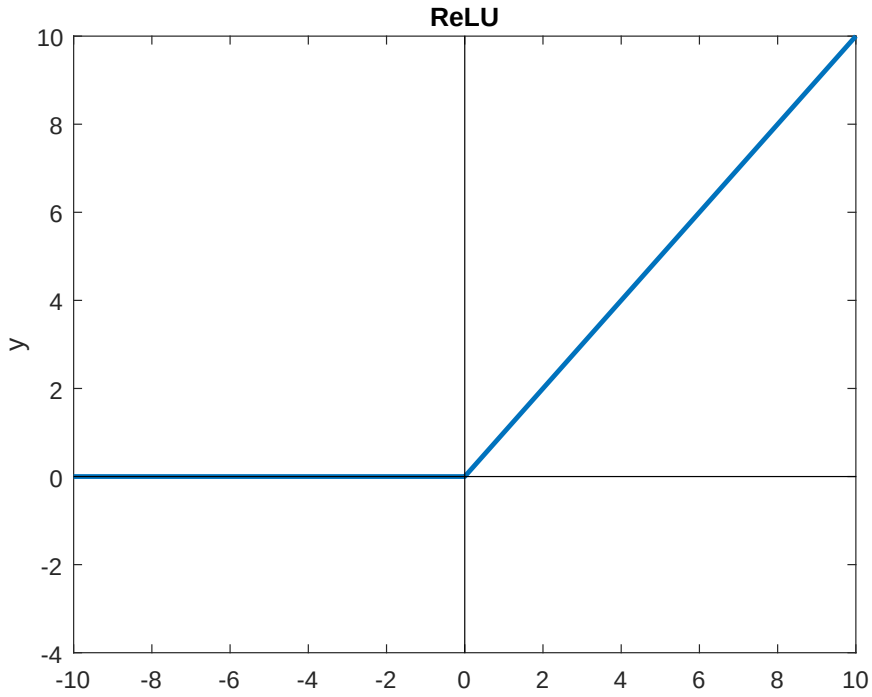


Figure 2.3: Output of ReLU

2.1.2 Pooling

Pooling is done to reduce the size of the representation. This means reducing the amount of parameters and computation deeper in the network, which is needed because the hardware used in this project will not have to handle the whole representation with pooling. It also serves to control overfitting. A downside is that the network becomes less sensitive to small variations in the input.

The operation is performed by taking a window which slides over the rectified feature map. One can use pooling types such as taking the maximum element, the average or the sum. See Figure 2.4 for an example with max pooling. Max pooling will be used in this project, because this appears to work best in practice [5].

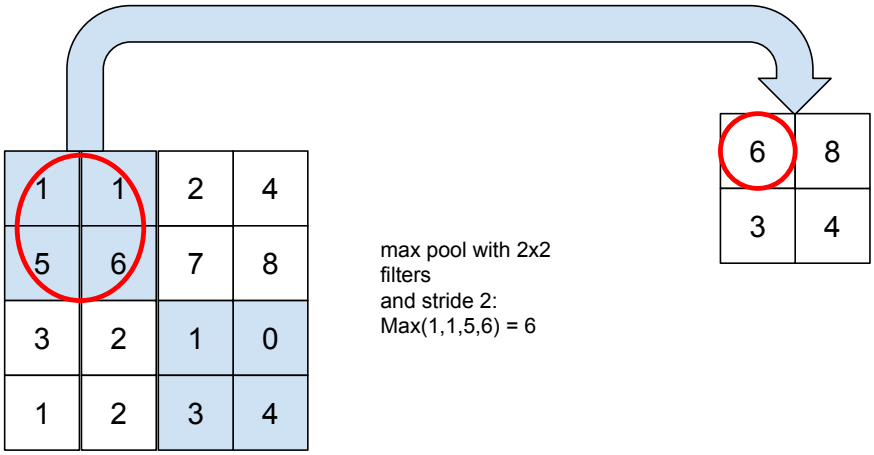


Figure 2.4: Max pooling

2.2 Autoencoder

This project deals with unlabeled training examples, which means using an unsupervised learning algorithm. In this case, an autoencoder is used, which is an unsupervised neural network (see Figure 2.5). Autoencoders are used to learn an approximation of the identity function in lower dimensional space. This means that if x is the input, the target value is also x. In other words, it tries to learn the identity function to get an output $\hat{x}$ that is very close to x. By putting constraints on the structure, one can discover correlations within the data. This is done by limiting the number of hidden units. [6] shows how autoencoders can be used in practice.

An autoencoder works like any neural network: the weights are randomly set at the start and when the output of the autoencoder does not match the input, weights are changed (often through backpropagation, see [7] for an example).

The autoencoder structure is used because the image set consists primarily of images without defects, so the network should generalize on these images and will be able to detect even the tiniest of defects.

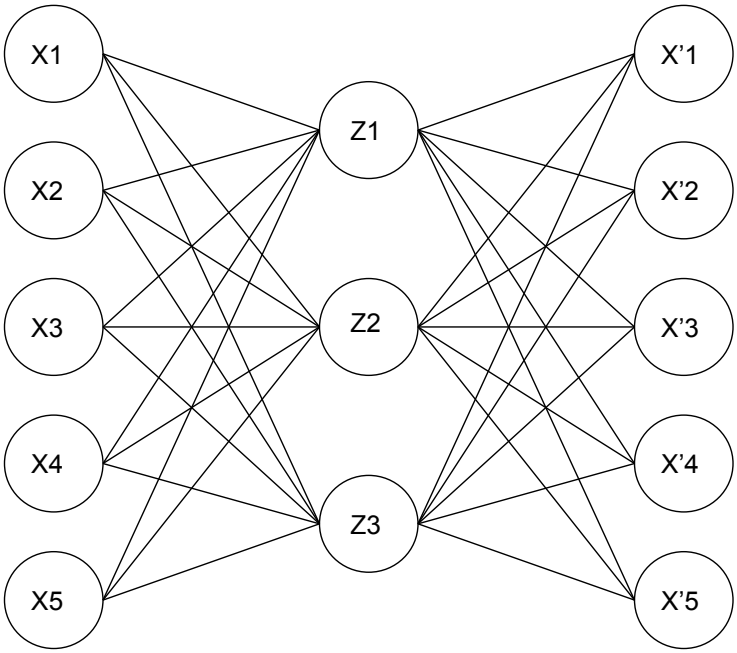


Figure 2.5: Simple autoencoder

2.3 Convolutional Autoencoder

Combining the autoencoder and CNN creates a convolutional autoencoder (CAE). A CAE carries out two additional operations: **unpooling** and **deconvolution**, so a CAE also has unpooling and deconvolutional layers. The number of unpooling/deconvolutional layers is the same as the number of convolutional/pooling layers. Before explaining the overall structure, these operations will be explained.

2.3.1 Unpooling

During pooling, the spatial size of the representation is reduced. As the name suggest, unpooling does the opposite. To unpool, the Kronecker product, denoted by $\otimes$, is used. The mathematical definition of the Kronecker product is [8]:
"Let A be an n by m matrix and b be a p by q matrix. Then the Kronecker product of A and B is the np by mq block matrix":

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \cdots & a_{nn}B \end{bmatrix}$$

Because max pooling is used, it is certain that the pixel values which were lost were equal or smaller to the output of max pooling. This means that the values of the second matrix used in the Kronecker product should never be bigger than 1. See Figure 2.6 for an example of unpooling. In this example, the second matrix in the Kronecker product only has value 1 in the upper left corner. Using a matrix of ones (uniform unpooling) would also be correct. In this project, uniform unpooling is used for the experiment. If average pooling would be used, every window of the unpooling operation output should have an average equal to the input of the unpooling operation. This means that the second matrix could, for example, be a simple matrix consisting of ones.

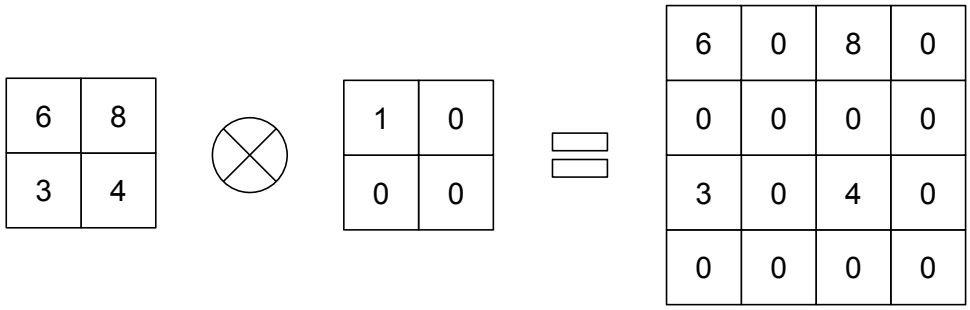


Figure 2.6: Unpooling: Kronecker product

2.3.2 Deconvolution

As the name suggests, this operation undoes the convolution operation. The name comes from [9]. However, the term deconvolution is not the best term because it is actually the transpose gradient of convolution. This means that when the gradient of the output d_{out} of convolution on an input x and filter w is given, the gradients of x and w (d_x and d_w respectively) have to be found. d_x can be found using the normal convolution operation (equation (1)) with input d_{out} and the transpose of w as filter. d_w can be found using equation (1) using the original input x and d_{out} as filter. With d_w and d_x, the output of the deconvolutional layer is calculated. See Figure 2.7 for an example of deconvolution.

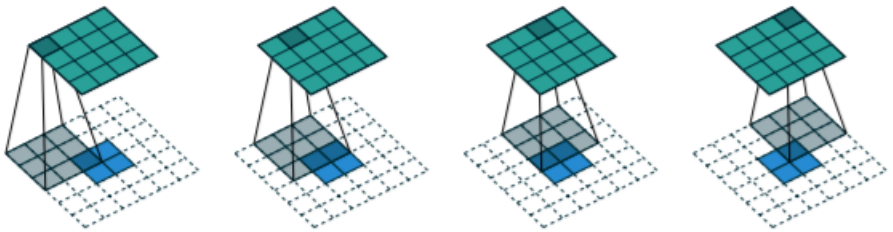


Figure 2.7: Deconvolution: the blue squares represent the input. The gray squares are the output. [10]

2.3.3 Overall Structure

The structure of a CAE is similar to that of a CNN. The encoding layer is the first half of the structure: the convolutional layers and the pooling layers together with a part of the fully connected layers. The middle of the fully connected layers is called the code. The other half of the fully connected layers, together with the unpooling and deconvolutional layers, form the decoding layer. Figure 2.8 shows the structure of the CAE.

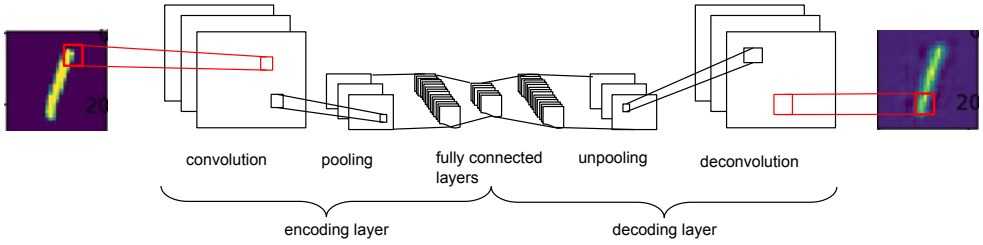


Figure 2.8: Convolutional autoencoder

Detecting anomalies in the data is done by first training the CAE. Let v and $\hat{v}$ be the input and output respectively. Then y is the absolute pixelwise difference between v and $\hat{v}$ [6], also known as the cost or the loss, see Figure 2.9. The value of y can then be used to check whether there is a significant difference between the expected output and the output given. This y is also known as the absolute residual.

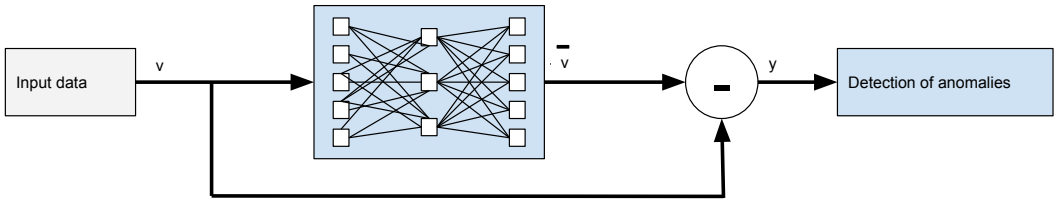


Figure 2.9: Detecting anomalies

Chapter 3

Methods

In this chapter the methods used in this project are elaborated upon. First, the software used will be discussed in Section 3.1. The MNIST image set [11] and CIFAR-10 image set [12] are used for a proof of concept. These datasets will be presented in Sections 3.2 and 3.3. In Section 3.4, the network structure and used settings are stated. Lastly, in Section 3.5, the validation of the network is explained.

3.1 TensorFlow

The network will be designed and implemented in TensorFlow [13]. TensorFlow is an open-source software library for Python which is written specifically for machine learning and neural networks, but can be used for other purposes as well. TensorFlow programs often have two sections: one section for building the network, the other section trains the network. Within the library, functions like convolution, pooling and deconvolution are present.
TensorFlow provides some forms of parallelism. TensorFlow uses nodes in a graph to represent mathematical operations, while the edges represent the data. When two nodes are not connected in the graph, these will be executed in parallel. For a CAE, parallelism means that a lot of computations can be done at the same time, which is benificial to the execution speed.

3.2 MNIST

The MNIST image set consists of 55,000 training examples of handwritten digits [11]. The training examples all have the same shape: 28 by 28 pixels. All images are grayscale, meaning that pixel values range from 0 to 1, where 0 is white and 1 is black. The MNIST image set also contains a validation set of 5,000 examples and a test set of 10,000 examples.

3.3 CIFAR-10

The CIFAR-10 image set consists of 50,000 training examples [12]. It also contains a test set of 10,000 examples. There are ten classes, which are listed in Figure 3.1. The images have a size of 32 by 32 pixels and are coloured. Because these images are used to simulate defects in the MNIST image set, the images will first have to be converted to grayscale and every image has to be cropped. To crop, the center 28 by 28 pixels are taken.

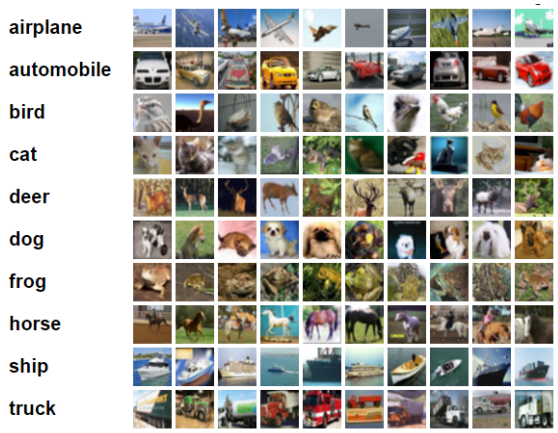


Figure 3.1: All CIFAR-10 classes

3.4 Network Structure

As in Figure 2.8, a convolutional autoencoder first has one or more convolutional and pooling layers, after which the output produced by these layers is used as input for the fully connected layers. The specific settings are as follows:

- Input layer: 28 by 28 pixels
- First convolutional layer: ten filters, size 5 by 5, strides [1, 1, 1, 1]
- First maxpooling layer: 4 by 4 window, strides [1, 4, 4, 1]
- Fully connected layers: $490 \rightarrow 245 \rightarrow 100 \rightarrow 245 \rightarrow 490$
- First unpooling layer: 4 by 4 uniform
- First deconvolutional layer: transpose of weights used in first convolutional layer
- Output layer: 28 by 28 pixels

Because of the limited time available, the optimal batch size and optimal stochastic optimization method will not be determined. According to [14], "when using a larger batch there is a degradation in the quality of the model, as measured by its ability to generalize". This is why a batch size of 32 is used for the proof of concept. The optimizer used is the Adam optimizer [15]. The default settings will be used.

3.5 Training and Validating the Network

3.5.1 Training

There are 55,000 training examples in the MNIST train image set. To simulate anomalies in the sewer pipe image set, 1% of the images of the training image set are randomly selected and replaced with CIFAR-10 images. This set is then used to train the network. Training is stopped when the network is converged. A network is considered as converged when the average cost of the current epoch (one full training cycle on the training set) is 0.01% above or below the average cost of the last epoch.

To see what the images look like after reconstructing with the trained network, ten images are selected from both the CIFAR-10 image set and the MNIST image set. The results are shown in Figure 3.2 and 3.3.

Figure 3.2: Reconstructed MNIST images

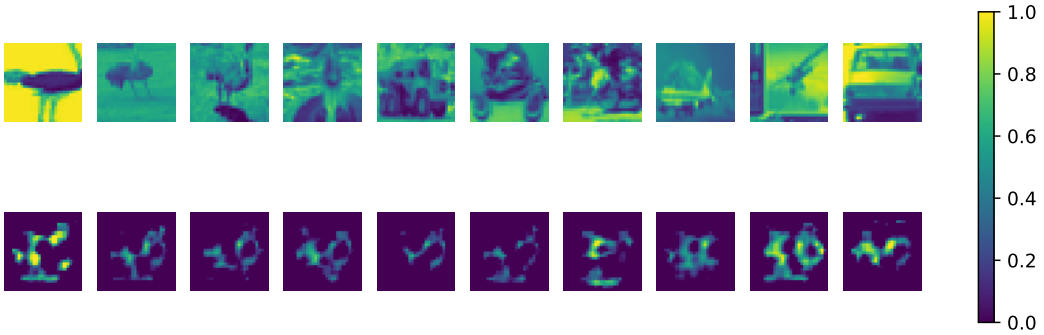


Figure 3.3: Reconstructed CIFAR-10 images

3.5.2 Validation

MNIST vs. CIFAR-10

To validate the network, the overlap of two boxplots is inspected: one which represents the cost per image of the MNIST images used for training and one which represents the cost per image of the CIFAR-10 images

used for training, see Figure 3.4. From this boxplot, it is clear that only the outliers from MNIST have some overlap with the boxplot representing the costs of the CIFAR-10 images.

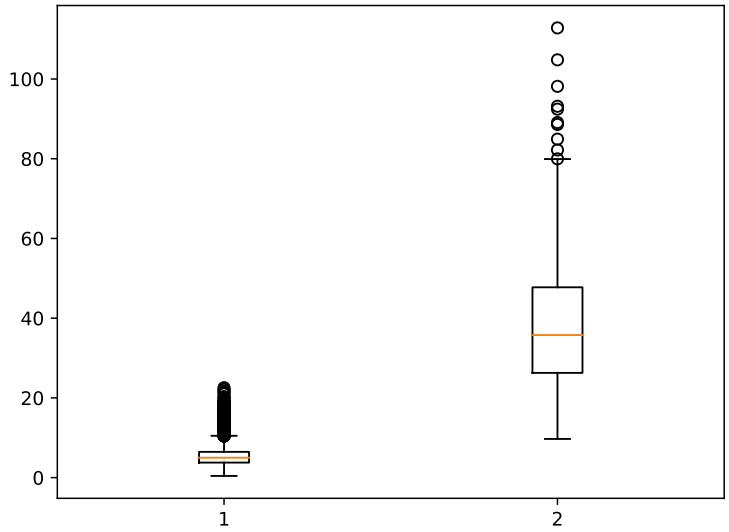


Figure 3.4: Boxplot 1: cost per image of MNIST. Boxplot 2: cost per image of CIFAR-10

Mann-Whitney U Test Preparation

To determine whether the cost of a randomly selected sample of the MNIST image set differs from the cost of a randomly selected sample of the CIFAR-10 image set, a statistical test has to be performed. One of these tests is the Mann-Whitney U test [16].

Before executing the test, a null hypothesis and alternative hypothesis have to be determined. When these are known, the α (significance level) has to be set. This α sets the range in which the null hypothesis is not rejected. The range is determined by looking up the corresponding z-value in a z-score table.
To execute the Mann-Whitney U test, the data has to be ordered. There are 54,450 samples of the MNIST image set and 550 of the CIFAR-10 image set: this was the training data. This data is put in one set and sorted on the cost from low to high. Then the data is ranked from lowest value to highest value, with the lowest value being rank 1 and the highest value being rank 55,000 (the number of costs in our set). When there are values in the set that are equal to 4 decimal numbers, the ranks are summed up and the average is taken. This average is the rank given to these values. An example:
Lets say there are four samples with the same cost of 5 in the set and they are on rank 5, 6, 7 and 8. This is corrected by summing the ranks, so every sample gets 6.5 as corrected rank. The value on the ninth spot still gets rank 9 if this sample has a unique value for the cost.
The U-values can be calculated with:

$$U_1 = n_1 n_2 + \left(\frac{n_1(n_1+1)}{2}\right) - R_1 \quad (2)$$

$$U_2 = n_1 n_2 + \left(\frac{n_2(n_2+1)}{2}\right) - R_2 \quad (3)$$

where

- n_1 = number of samples in the MNIST set
- n_2 = the number of samples in the CIFAR-10 set
- R_1 = the sum of ranks of the MNIST set
- R_2 = the sum of ranks of the CIFAR-10 set

From these U-values, the lowest score is taken. Because this is a rather large dataset, one can assume that U is approximately normally distributed. With this assumption, the standardized value z is calculated:

$$z = \frac{U \cdot \mu}{\sigma} \quad (4)$$

where μ is the mean of U and σ is the standard deviation of U. μ is given by:

$$\mu = \frac{n_1 n_2}{2} \quad (5)$$

σ is given by:

$$\sigma = \sqrt{\frac{n_1 n_2 (n_1 + n_2 + 1)}{12}} \quad (6)$$

Executing the Mann-Whitney U Test

To execute the test, the following steps are performed:

1. Define null and alternative hypothesis
2. State alpha
3. State decision rule
4. Calculate test statistic
5. State results
6. State conclusions

Lets start with the first step.

Define null and alternative hypothesis:
H_0: There is no difference between the medians of the reconstruction costs of the two image sets.
H_1: There is a difference between the medians of the reconstruction costs of the two image sets.

State alpha:

$\alpha = 0.05$

State decision rule:

From α, the z-value used is 1.96 for this test. This means that if the z-value is less than -1.96 or bigger than 1.96, the null hypothesis is rejected.

Calculate test statistic:

After ordering the data and correcting the ranks, the values R_1 and R_2 are:

$$R_1 = 1,482,466,588$$
$$R_2 = 30,060,912$$

With these values, the U-values are determined using equations (2) and (3):

$$U_1 = 54,450 \cdot 550 + (\tfrac{54,450(54,450+1)}{2}) - 1,482,466,588 = 29,909,387.5$$
$$U_2 = 54,450 \cdot 550 + (\tfrac{550(550+1)}{2}) - 30,060,912 = 38,112.5$$

Now the z-value (equation (4)) is calculated with U_2, our lowest U-value:

$$z = \frac{38,112.5 - \frac{54,450 \cdot 550}{2}}{\sqrt{\frac{54,450 \cdot 550(54,450+550+1)}{12}}} = -40.31$$

State results

If z is less than -1.96 or greater than 1.96, reject the null hypothesis. z is equal to -40.31, so less than -1.96. This is equivalent to $p < 10^{-5}$. The null hypothesis is therefore rejected.

State conclusions

The null hypothesis is rejected, which means that the median of the MNIST set differs from the median of the CIFAR-10 set. The Area Under Curve (AUC) can be calculated using

$$\text{AUC} = \frac{U}{n_1 n_2}$$

The AUC is used here as measure of performance. The AUC is equal to 0.999 which is almost equal to 1. This means the CAE has a good performance.

In Conclusion

The network can detect anomalies when the MNIST train image set is used in combination with 550 CIFAR-10 images, where CIFAR-10 images are seen as anomalies. In the next chapter, the sewer pipe image sets will be used to check whether the CAE can also be used to detect anomalies in the sewer pipes image sets.

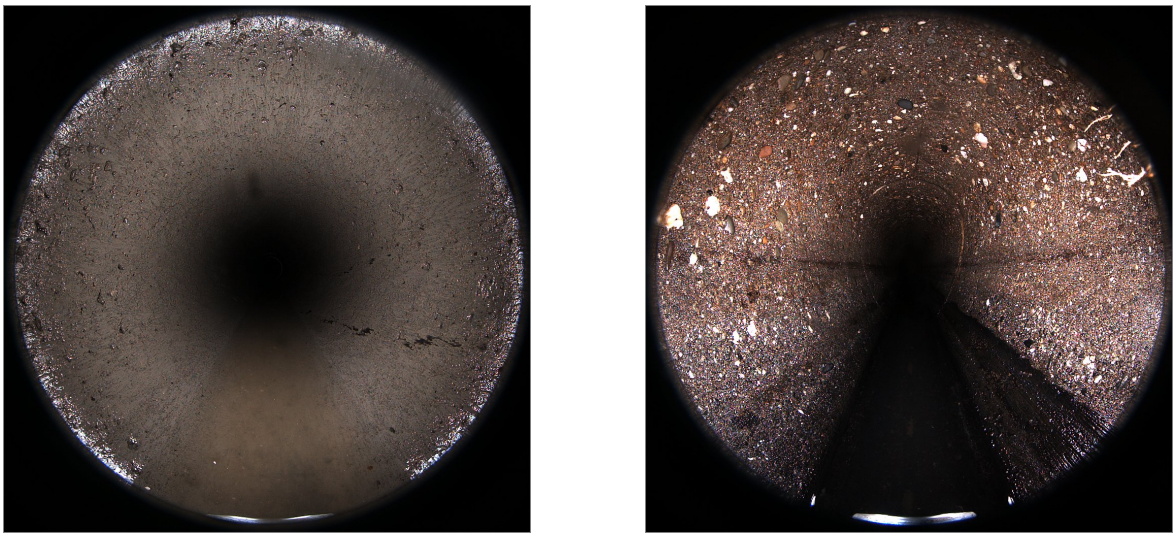
Chapter 4

Experiment

In this chapter, the experiment on the sewer pipe image sets will be explained. First, in Section 4.1, the available data will be discussed. In Section 4.2, the network structure that will be used is explained. Lastly, there is an explanation with regard to how the performances of the CAEs are validated in Section 4.3.

4.1 Sewerpipe Image Sets

In this project, two sewer pipe image sets were used to experiment on. These two sets correspond to two different types of pipe: smooth concrete and coarse agglomerate, see Figure 4.1 below.



(a) Image from first set

(b) Image from second set

Figure 4.1: Images from the two different sets

Because of the difference, two seperate CAEs will be trained: one for the first image set and one for the second image set. From now on, the first image set will be refered to as "smooth image set" and the second image set

will be refered to as “coarse image set”.

4.1.1 Available data

The first image set consists of 684 images, the second consists of 698 images. The images are 1040 pixels by 1040 pixels and the images are colored (RGB). All images are aligned. For validation, there are small labeled subsets of eleven images available for both the first and the second image set. In this subset, every image is divided into 324 patches of 40 by 40 pixels. Every patch is labeled as “normal” or as “anomaly”. When labeled as anomaly, this does not necessarily mean that there is a defect: it could, for example, be a connection with another pipe or different lighting. See Figure 4.2 for an example.

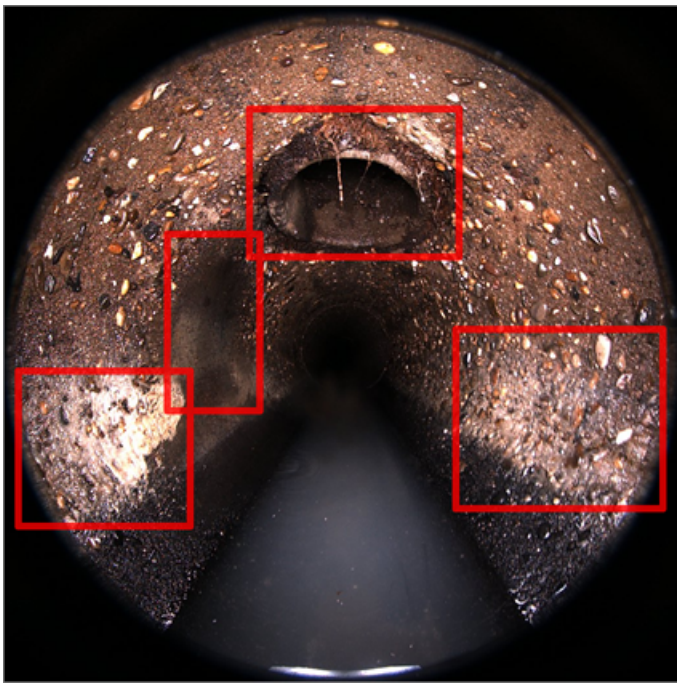


Figure 4.2: Regions of interest

The patches are used because, as seen in Figure 4.2, the images are always black in the middle and at the edges. Checking these parts on anomalies is not interesting, so these parts are discarded. The patches are determined by calculating the distance between the center of the image and the coordinates. Figure 4.3 shows the 324 patches used.

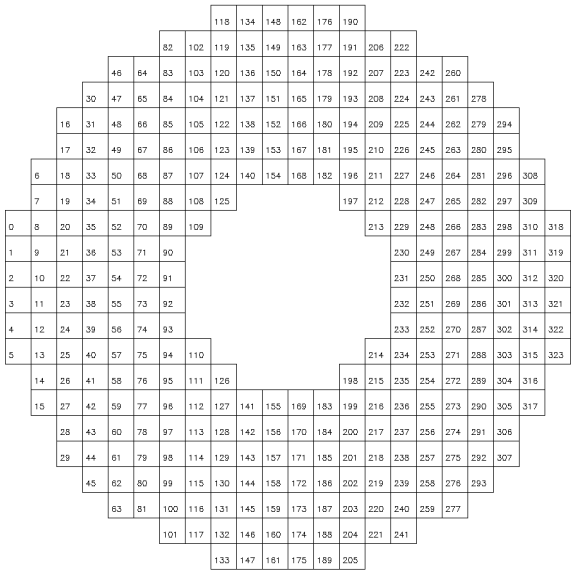


Figure 4.3: All 324 patches

4.2 Network Structure

For the experiment, a total of two CAEs will be trained: one CAE per image set.
The network structure will have two convolutional layers and two maxpooling layers in the encoder part, plus two fully connected layers, followed by the code (middle fully connected layer). This means the decoder will also have two fully connected layers, followed by two unpooling layers and two deconvolutional layers. The maxpool window will be small, because according to [17], the window size that will be used performs well. The specific settings are as follows:

- Input layer: 1040 by 1040 by 3 pixels
- First convolutional layer: ten filters, size 20 by 20 by 3, strides [1, 10, 10, 1]
- First maxpooling layer: 2 by 2 window, strides [1, 2, 2, 1]
- Second convolutional layer: ten filters, size 20 by 20 by 3, strides [1, 2, 2, 1]
- Second maxpooling layer: 2 by 2 window, strides [1, 2, 2, 1]
- Fully connected layers: $1690 \rightarrow 845 \rightarrow 422 \rightarrow 845 \rightarrow 1690$
- First unpooling layer: 2 by 2 uniform
- First deconvolutional layer: transpose of weights used in second convolutional layer
- Second unpooling layer: 2 by 2 uniform
- Second deconvolutional layer: transpose of weights used in first convolutional layer
- Output layer: 1040 by 1040 by 3 pixels

4.3 Validation

To see how the CAEs perform, the two labeled subsets are used. The cost per patch will be calculated, with the cost being the absolute difference between the input and the output. With the costs, a Mann-Whitney U test will be performed as explained in Section 2.5.2 and an ROC-curve will be generated.

Chapter 5

Results

In this chapter the results of the experiments described in Chapter 4 are presented. The results of the smooth image set are presented in Section 5.1, the results of the coarse image set are presented in Section 5.2. Section 5.3 compares the reconstructed images and difference images from the smooth set and the coarse set.

5.1 Smooth Image Set

Below, the Mann-Whitney U test results are stated. For the smooth image set, there are 2763 patches labeled "normal" and 801 patches labeled "anomalous".

Define null and alternative hypothesis:
H_0: There is no difference between the medians of the reconstruction costs of the set patches labeled "normal" and the set patches labeled "anomalous".
H_1: There is a difference between the medians of the reconstruction costs of the set patches labeled "normal" and the set patches labeled "anomalous".

State alpha:
$\alpha = 0.05$

State decision rule:
From α, the z-value used is 1.96 for this test. This means that if the z-value is less than -1.96 or bigger than 1.96, the null hypothesis is rejected.

Calculate test statistic:
After ordering the data and correcting the ranks, the values R_1 and R_2 are:

$$R_1 = 3,937,726.5$$
$$R_2 = 2,415,103.5$$

With these values, the U-values are determined:

$$U_1 = 2,763 \cdot 801 + (\tfrac{2,763(2,763+1)}{2}) - 3,937,726.5 = 2,093,902.5$$
$$U_2 = 2,763 \cdot 801 + (\tfrac{801(801+1)}{2}) - 2,415,103.5 = 119,260.5$$

Now the z-value is calculated with U_2, our lowest U-value:

$$z = \frac{119,260.5 - \frac{2,763 \cdot 801}{2}}{\sqrt{\frac{2,763 \cdot 801(2,763+801+1)}{12}}} = -38.50$$

State results

If z is less than -1.96 or greater than 1.96, reject the null hypothesis. z is equal to -38.50, so less than -1.96. This is equivalent to $p < 10^{-5}$. The null hypothesis is therefore rejected.

State conclusions

The null hypothesis is rejected, which means that the median of the reconstruction costs of the patches labeled "normal" is less than the median of the reconstruction costs of the patches labeled "anomalous". The AUC is equal to 0.9461, as can be seen in the ROC-curve in Figure 5.1.

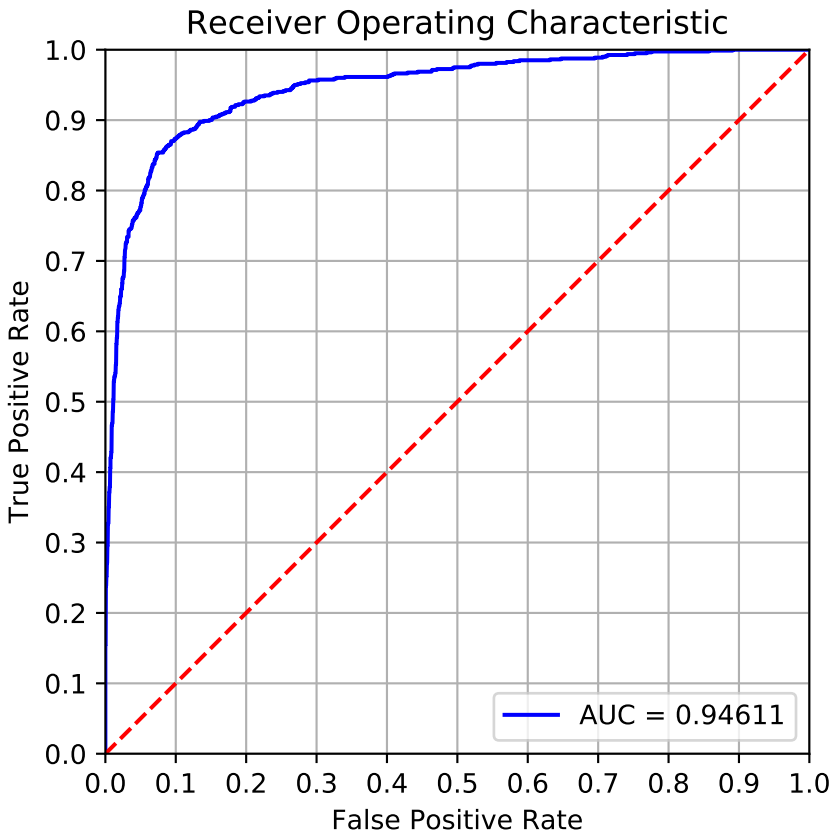


Figure 5.1: ROC-curve of labeled smooth image subset

Looking at the result of the Mann-Whitney U test, one can see that the CAE is able to detect most of the regions of interest. The mean absolute difference is used to obtain the results shown in Figure 5.1. Whether the performance is high enough depends on what the acceptable sensitivity (true positive rate) is. For example, when a sensitivity of 0.9 is wanted, there will be a false positive rate of 0.14, which means that 14% of the negatives would be considered positive by the CAE. However, the AUC is high enough to be of use.

5.2 Coarse Image Set

For the coarse image set, there are 2758 patches labeled "normal" and 806 patches labeled "anomalous".

Define null and alternative hypothesis:
H_0: There is no difference between the medians of the reconstruction costs of the set patches labeled "normal" and the set patches labeled "anomalous".
H_1: There is a difference between the medians of the reconstruction costs of the set patches labeled "normal" and the set patches labeled "anomalous".

State alpha:
α = 0.05

State decision rule:
From α, the z-value used is 1.96 for this test. This means that if the z-value is less than -1.96 or bigger than 1.96, the null hypothesis is rejected.

Calculate test statistic:
After ordering the data and correcting the ranks, the values R_1 and R_2 are:

$$R_1 = 4,438,165$$
$$R_2 = 1,914,665$$

With these values, the U-values are determined:

$$U_1 = 2,758 \cdot 806 + (\tfrac{2,758(2,758+1)}{2}) - 4,438,165 = 1,589,444$$
$$U_2 = 2,758 \cdot 806 + (\tfrac{806(806+1)}{2}) - 1,914,665 = 633,504$$

Now the z-value is calculated with U_2, our lowest U-value:

$$z = \frac{633,504 - \frac{2,758 \cdot 806}{2}}{\sqrt{\frac{2,758 \cdot 806(2,758+806+1)}{12}}} = -18.60$$

State results

If z is less than -1.96 or greater than 1.96, reject the null hypothesis. z is equal to -18.60, so less than -1.96. This is equivalent to $p<10^{-5}$. The null hypothesis is therefore rejected.

State conclusions

The null hypothesis is rejected, which means that the median of the reconstruction costs of the patches labeled "normal" is less than the median of the reconstruction costs of the patches labeled "anomalous". The AUC is equal to 0.715, as can be seen in the ROC-curve in Figure 5.2.

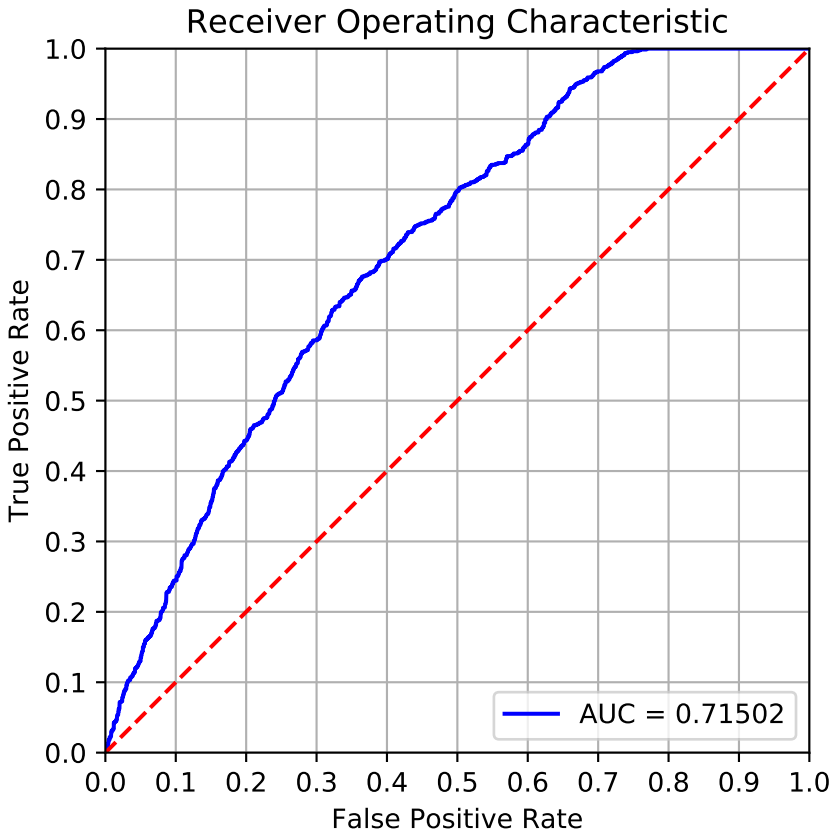


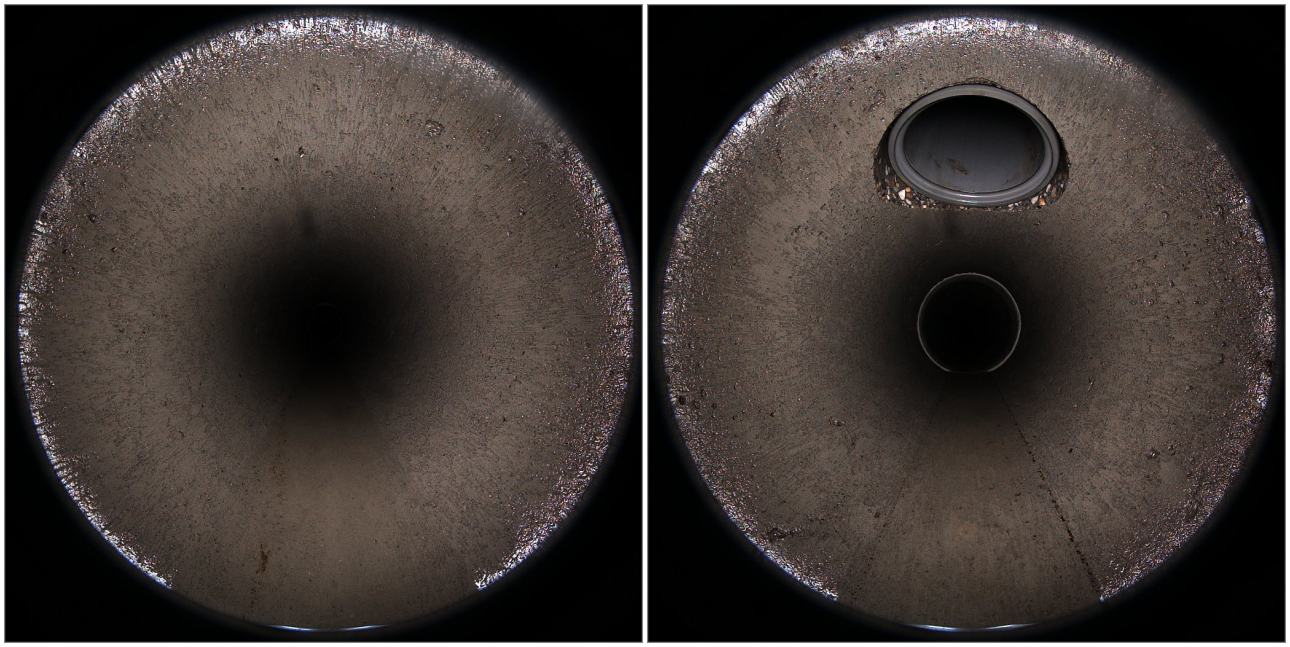
Figure 5.2: ROC-curve labeled coarse image subset

In this case, the performance of the CAE is clearly worse than the performance of the CAE for the smooth image set. The Mann-Whitney U test shows that the CAE is able to detect a significant difference between patches labeled as "normal" and patches labeled as "anomalous", but the difference is not as noticeable as with the smooth image set. If the system were to be calibrated to an acceptable sensitivity, there would still be a large a large amount of false positives. The AUC is quite low: the CAE does not have a respectable performance.

The difference in performance of the CAE on the smooth image set and on the coarse image set is caused by the texture of the concrete in the coarse image set. Besides this difference, the smooth sewer pipes also have approximately the same color concrete across the whole pipe, while the coarse sewer pipes contain more variety in color, which is also a cause for the performance difference.

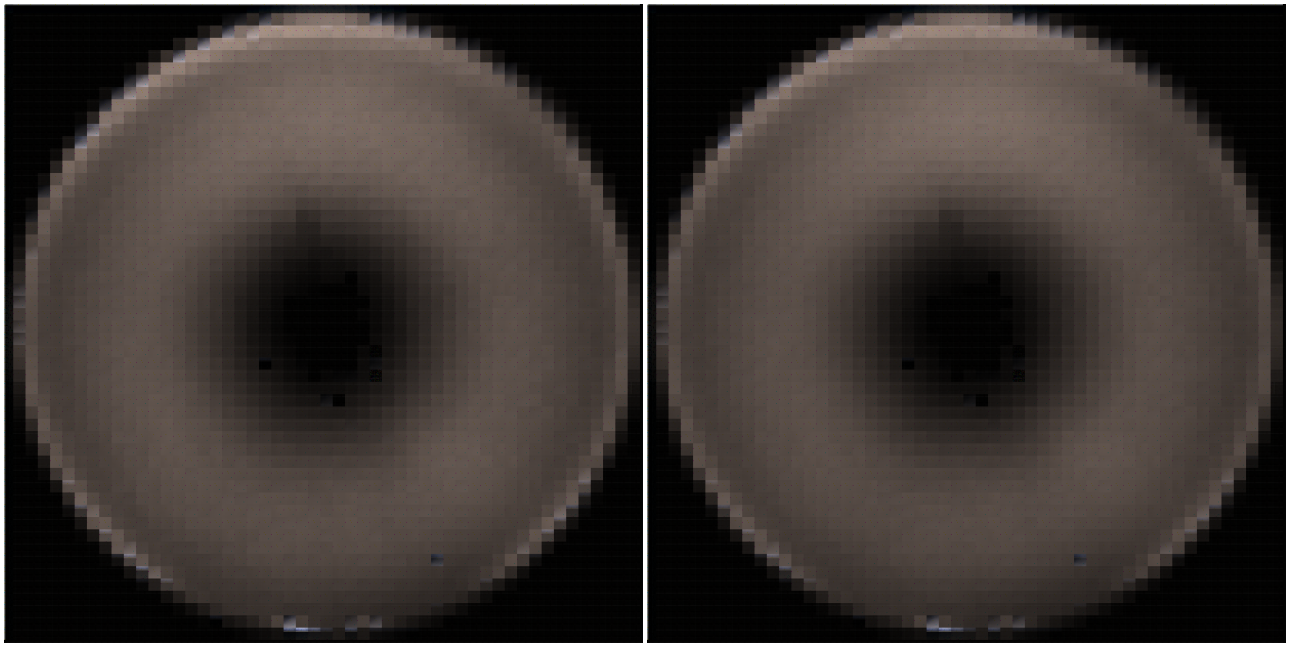
5.3 Comparing Smooth and Coarse

To see what the difference is in how the CAE reconstructs the images, two images from both sets are taken. Their difference image is also presented. The input images from the smooth image set are shown in Figure 5.3. The reconstructed images are presented in Figure 5.4 and the difference images are presented in Figure 5.5.



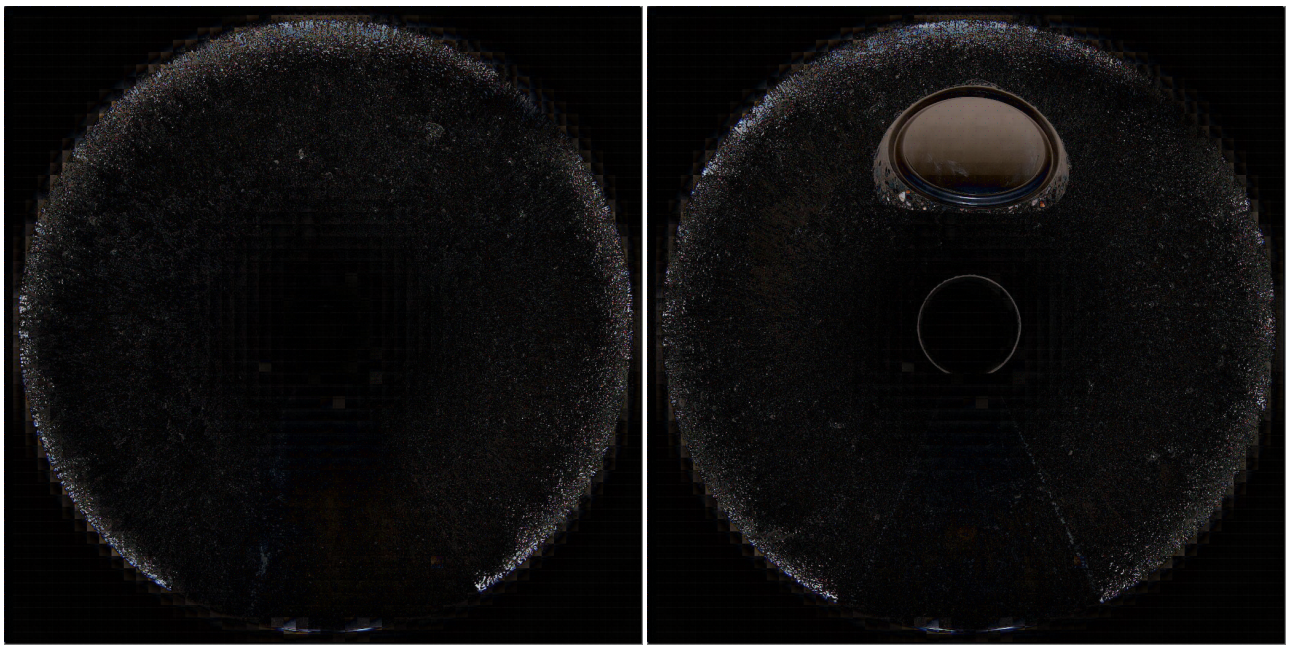
(a) First image (b) Second image

Figure 5.3: Input images



(a) Reconstruction of Figure 5.3a (b) Reconstruction of Figure 5.3b

Figure 5.4: Reconstructed images

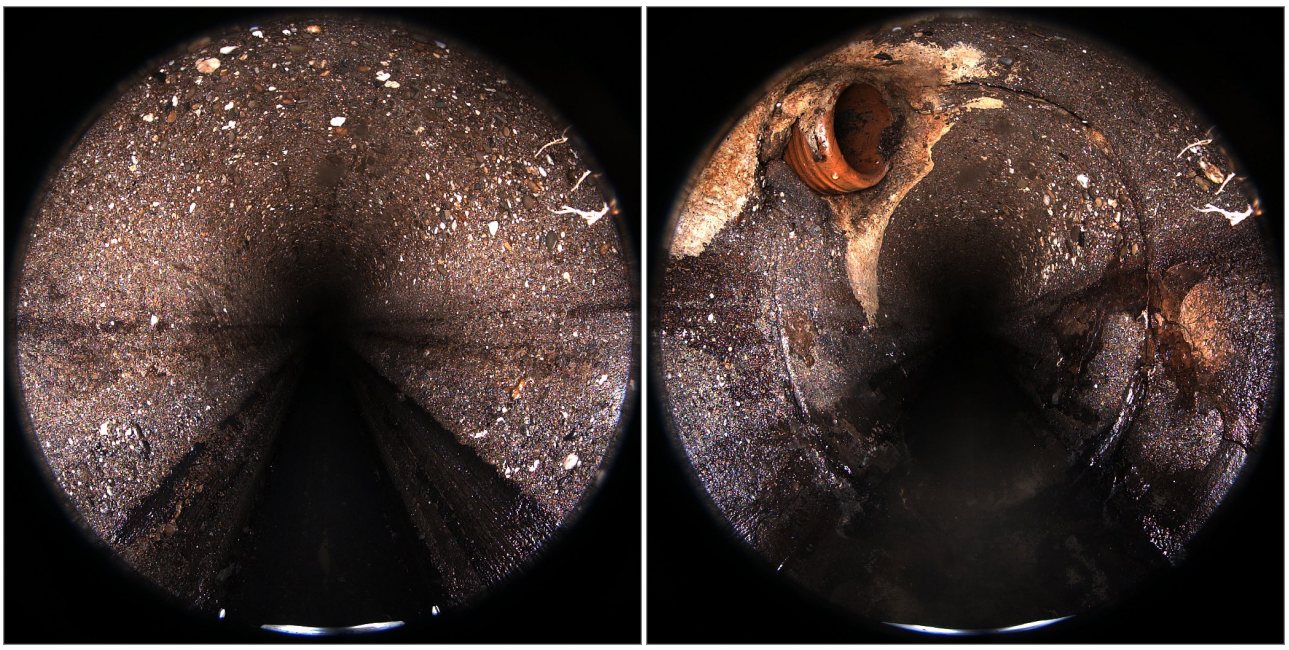


(a) Difference of Figure 5.3a (b) Difference of Figure 5.3b

Figure 5.5: Difference images: absolute difference between Figure 5.3 and Figure 5.4

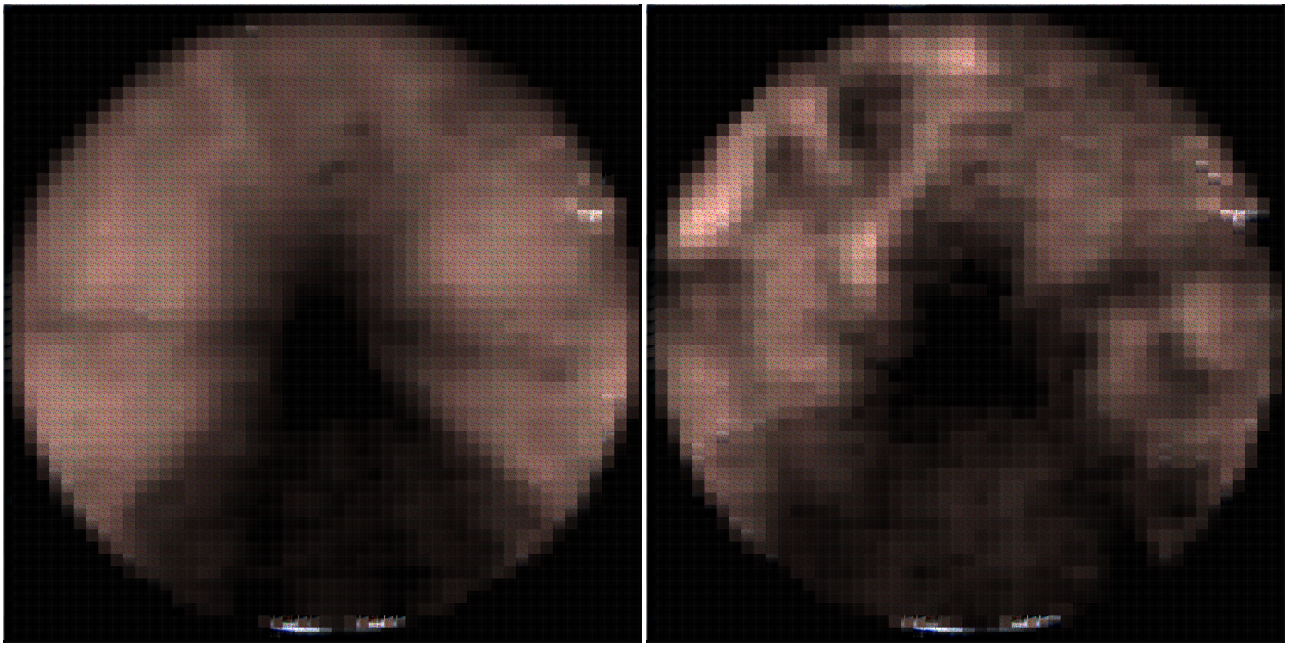
From the reconstructed images and difference images, it is clear that the CAE is able to detect regions of interest: Figure 5.4b does not show the connection pipe at the top and Figure 5.5b shows that there is a clear difference at the position of the connection pipe. This means that it is very likely that the cost in the patches covering the position of the connection pipe is higher than normal. Figure 5.5a does not show any big differences, as Figure 5.3a does not have any big regions of interest.

Now the coarse image set: the input images are shown in Figure 5.6. The reconstructed images are presented in Figure 5.7 and the difference images are presented in Figure 5.8.



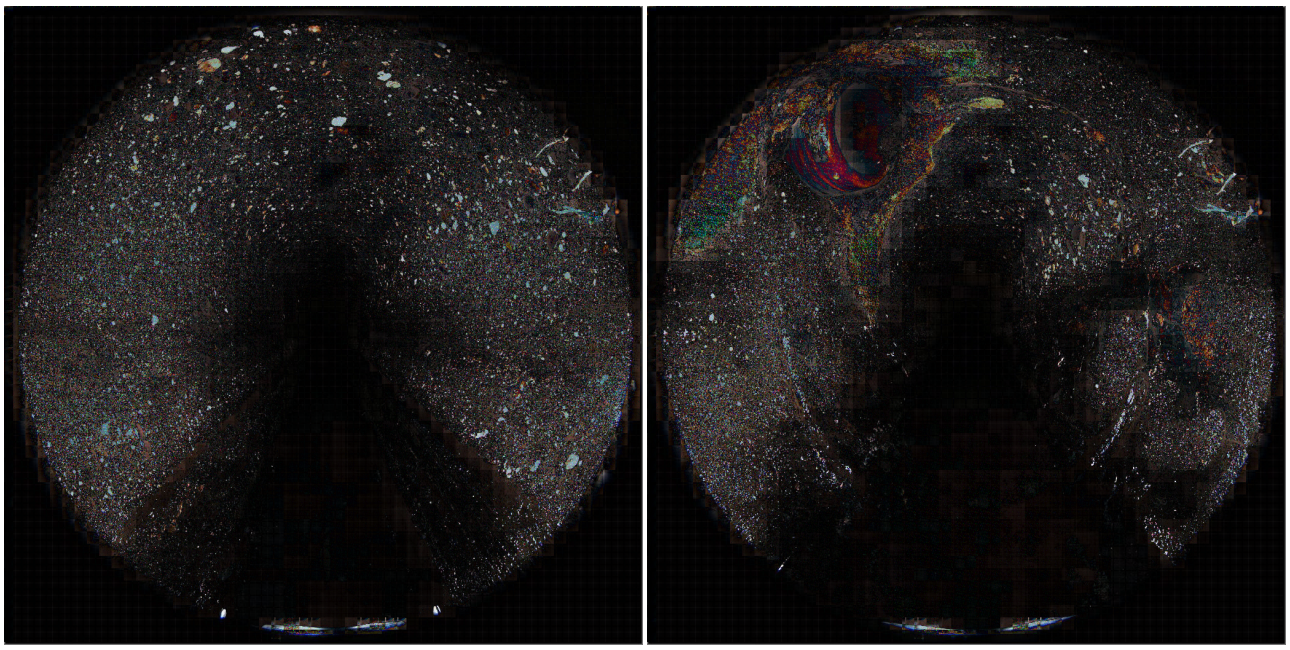
(a) First image (b) Second image

Figure 5.6: Input images



(a) Reconstruction of Figure 5.6a (b) Reconstruction of Figure 5.6b

Figure 5.7: Reconstructed images



(a) Difference of Figure 5.6a (b) Difference of Figure 5.6b

Figure 5.8: Difference images: absolute difference between Figure 5.6 and Figure 5.7

It is clear that the CAE performs worse on the coarse image set. Figure 5.8a shows that the CAE is not able to train on the texture present on the surface of the pipe: the top of the difference image shows the texture which is present in Figure 5.6a. This means that the cost in the patches covering the top might be too high: the CAE might rank these patches as high as other regions which are true regions of interest, while these are not.

Figure 5.6b shows a connection pipe at the top left corner. As in Figures 5.3b and 5.4b, this should not be visible in the reconstruction of the image. However, the connection pipe is still somewhat visible in Figure 5.7b. In this case, the difference image in Figure 5.8b shows the connection pipe quite clearly, but in other similar cases this might cause the cost in the patches covering the region of interest to be too low: the CAE might rank these patches lower than other patches not containing a true region of interest.

Chapter 6

Conclusion

6.1 Summary

The research question for this project was:
Is the use of convolutional autoencoders viable for detecting anomalies in aligned image sets?
This was done by first analysing the performance of a convolutional autoencoder on an image set consisting of 54,450 MNIST images and 550 CIFAR-10 images. The CIFAR-10 images were used to simulate defects in the image set in order to simulate the sewer pipe image sets. The performance of the convolutional autoencoder was excellent: the AUC was almost equal to 1.

After this proof of concept, a convolutional autoencoder was designed for the application in sewer pipe image sets. Analysis showed that the convolutional autoencoder for the smooth image set had an acceptable performance: the AUC was equal to 0.946 for the smooth image set. However, the AUC for the coarse image set was equal to 0.715 which is noticeably worse than the performance on the smooth image set. The ROC-curve showed that, when the system is calibrated to an acceptable sensitivity, the amount of false positives will be large. The following reasons might be the cause of this difference in performance:

- The convolutional autoencoder has more difficulty training on the coarse image set due to the texture on the surface of the sewer pipes and the color variety that is present in the coarse image set.
- Optimizing the parameters is barely done. This means that the settings used in this project might be far from optimal. Using different settings could result in a significant performance gain for both the smooth image set and coarse image set.

In conclusion: from this research, there is no straightforward answer to the research question as posed in the beginning of this chapter. The design that was used in this project shows that for the smooth image set the use of convolutional autoencoders is a viable option. However, for the coarse image set, the settings used were not sufficient. To rule the use of convolutional autoencoders out completely, more research has to be performed.

6.2 Future Work

For future research, the following may be investigated.

Optimizing network: There is a large chance that the design of the network used in this project is not optimal. An improved design could give a big performance boost.

Using reconstructed features instead of reconstructed images: Using a convolutional autoencoder is both advantageous and disadvantageous. An advantage is that one does not need to have specific domain knowledge. A disadvantage is that it is unavoidable that the reconstructed images are compared with the input image, because it is not known what the features should be beforehand. Using principal component analysis (PCA), a feature descriptor is used. This feature descriptor can be chosen on basis of what type of abnormal region has to be detected. For example, an identity function could be used or a feature that performs well on edge detection. After choosing the feature descriptor, the features are used in a PCA, where the features are first decomposed and then (partially) reconstructed. Thus, this outputs a reconstructed feature in stead of a reconstructed image. Figure 6.1 shows the PCA framework.

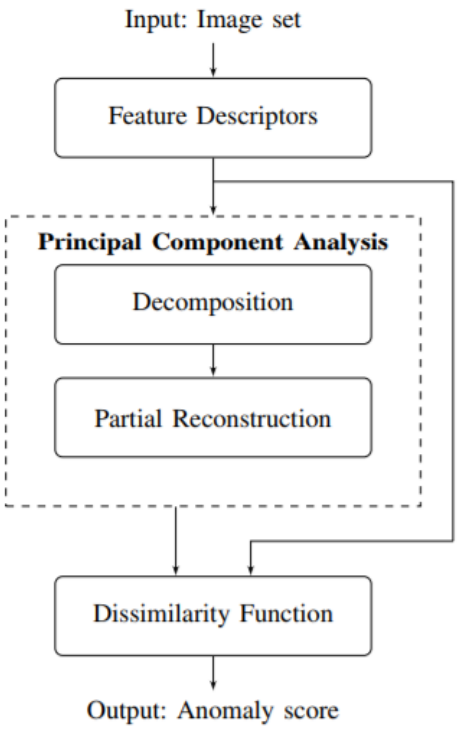


Figure 6.1: PCA framework

In a research paper presented in Appendix A, research was performed on the same sewer pipe image sets using the PCA-based method. This method had approximately the same performance on the smooth image set. However, on the coarse image set, the AUC was 0.818, which is noticeably better than the performance of the convolutional autoencoder on the coarse image set. The difference in comparing reconstructed images

with input images versus comparing the reconstructed features with the original features might be the cause of the performance difference. For future research, it may be investigated whether the performance of the convolutional autoencoder improves if the unpooling and deconvolutional layers are disposed of. This would make the convolutional autoencoder method more similar to the PCA-based method.

Bibliography

[1] TTW NWO. Sewersense. https://www.universiteitleiden.nl/onderzoek/onderzoeksprojecten/wiskunde-en-natuurwetenschappen/sewersense, 2016. Accessed: 28-02-2017.

[2] Stanford. Cs231n: Convolutional neural networks for visual recognition. http://cs231n.github.io/, 2017. Accessed: 28-02-2017.

[3] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[5] Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR'10)*. IEEE, 2010.

[6] Marco Martinelli, Enrico Tronci, Giovanni Dipoppa, and Claudio Balducelli. Electric power system anomaly detection using neural networks. In *Knowledge-Based Intelligent Information and Engineering Systems, 8th International Conference, KES 2004, Wellington, New Zealand, September 20-25, 2004. Proceedings. Part I*, pages 1242–1248, 2004.

[7] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[8] Kathrin Schäcke. On the kronecker product. 2013.

[9] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2528–2535, June 2010.

[10] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *ArXiv e-prints*, 2016.

[11] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 2013. Accessed: 28-02-2017.

[12] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[13] Google Brain Team. Tensorflow: an open-source software library for machine intelligence. https://www.tensorflow.org/, 2015. Accessed: 28-02-2017.

[14] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836, 2016.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[16] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. 1947.

[17] Dmytro Mishkin, Nikolay Sergievskiy, and Jiri Matas. Systematic evaluation of CNN advances on the imagenet. *CoRR*, abs/1606.02228, 2016.

Appendix A

Scientific Research Paper

Parallel to this thesis, research was done with the same purpose but with a different method. The research paper presented here is the result of that research. A PCA-based method is used. For the smooth image set, the PCA-based method produced similar results as the convolutional autoencoder. However, for the agglomerate image set, the results were noticeably better.

Unsupervised Anomaly Detection in Sewer Images with a PCA-based Framework

Dirk Meijer†, Mitchell Kesteloo and Arno Knobbe
Leiden Institute for Advanced Computer Science, Leiden, the Netherlands
†Correspondence: meijerdwj@liacs.leidenuniv.nl

***Abstract*— We propose a simple framework for unsupervised anomaly detection in aligned image sets, consisting of (i) feature extraction, (ii) PCA decomposition and partial reconstruction, and (iii) a dissimilarity measure comparing reconstructed to extracted features. The basic principle is explained using artificial datasets, and we show the effectiveness in a real-world scenario pertaining to sewer inspection images. We investigate the effectiveness of several features for this specific task, and show that concatenating several features results in superior performance. An analogy is also drawn to convolutional autoencoders and we compare to some simplistic renditions of such networks to our framework.**

***Keywords*— Anomaly detection, Principal component analysis, Convolutional autoencoder, Unsupervised learning, Image processing, Sewer asset management**

I. Introduction

Sewer inspections need to be performed periodically to ensure performance is up to standards. Image and video data collected by a 'pipe inspection gadget' (PIG) is often manually inspected, leading to subjective, inconsistent, and unreliable deterioration and urgency ratings [1]. The SewerSense project [2] aims to automate such sewer inspections.

Anomaly detection (sometimes outlier or novelty detection) is a problem which aims to detect observations that do not conform to an expected pattern. In the context of image and video data, this relates to finding regions of interest (ROIs), portions of the video or images that have a different appearance and might be of interest. In the context of sewer inspections, automatic ROI detection is the first step of the SewerSense project. At a later stage in the project, classification of the found regions into a taxonomy of defect classes will be considered, but the intermediate result should aid the inspection process in itself.

In this work, we propose a three-part framework to detect anomalies in *aligned image sets*, such as static camera video or photographs, or registered images. The framework is based on principal component decomposition and partial reconstruction, but accounts for the fact that not all common elements in image sets can be accounted for by a linear model (such as PCA is) by first extracting possibly non-linear features from the image sets. We also foray into the field of deep learning and investigate the possibility of using convolutional autoencoders (CAEs) to fill the role of several parts of the framework.

We would like to emphasize that while this work originated from the need to automatically process sewer pipe images, no assumptions are made specific to this problem. The only requirement is that *the images in a set are aligned*, so other possible applications include video surveillance, autonomous vehicles and medical image processing.

II. Preliminaries

A. Principal Component Analysis

Principal Component Analysis (PCA) has been around for over a century, after having been introduced in 1901 by Karl Pearson [3]. It is a popular tool in statistics, data science and many other scientific fields, used to reduce the dimensionality of data to facilitate data exploration and the use of algorithms that are sensitive to high dimensionality.

Given a dataset $\mathbf{X}$, consisting of N observations with d features each, we express this as an $[N \times d]$ matrix. PCA is performed by calculating the covariance matrix $\mathbf{C}$ of this dataset and performing eigenvalue analysis on $\mathbf{C}$. This results in d eigenvalues and d eigenvectors (or principal components) of length d. These eigenvectors form an orthonormal basis for the space in which our dataset $\mathbf{X}$ exists and the corresponding eigenvalues are proportional in magnitude to the variance of dataset $\mathbf{X}$ explained by each eigenvector (their sum being equal to the total variance present in $\mathbf{X}$).

When using PCA for dimensionality reduction, a dimensionality $d_1 \leq d$ is chosen and $\mathbf{X}$ is projected[1] on the first d_1 eigenvectors (in order of descending eigenvalues). The projected $\mathbf{X}_p$ retains as much variance as is possible in d_1 dimensions[2]. This allows researchers to view high-dimensional data in two or three-dimensional plots, or employ algorithms that are not designed for high-dimensional data. When $d_1 = d$, we can return to the original space by inverting the projection matrix and adding the mean values of each feature after transformation, without any loss of information.

B. Anomaly Detection

In this study, we approach anomaly detection as an unsupervised learning problem. This means the algorithm will learn the structure present in the data, with the caveat that some parts of the data will not adhere to this structure. These anomalies have to be detected without influencing the learning

[1]The covariance matrix and the newly-found orthonormal basis do not contain the mean values of the original dataset $\mathbf{X}$, so the dataset should be centered around zero before projecting onto the basis.

[2]Barring non-linear embeddings.

of the structure too much. This draws some parallels to one-class classification and clustering. The difference in one-class classification is the assumption that the dataset used to learn the structure will be pure, that is to say, it does not contain the anomalies that are to be detected after learning [4]. The difference with clustering is that clustering cares about different classes of objects that may be present in the dataset, while anomaly detection only looks for the anomalies, regarding all non-anomalous objects as a single class. Anomalies might also be single instances, meaning they would not be considered as clusters.

Extensive work has been performed in the area of anomaly detection, but an especially well-written and extensive overview of methods, empirical issues, and literature is given in [5].

III. Framework

We propose a simple three-part framework to detect local anomalies in aligned image sets and videos, as shown in figure 1. The three parts consist of (i) feature descriptors, (ii) PCA decomposition and partial reconstruction, (iii) a dissimilarity function to compare the PCA reconstructed feature to the extracted features.

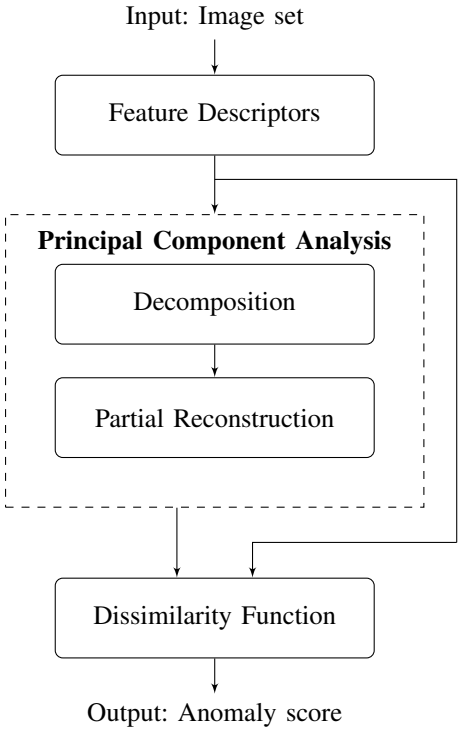


Fig. 1. The proposed three-part anomaly detection framework.

a) Feature Descriptors: The choice of feature descriptor depends on the type of anomaly that has to be detected in the images. For example, to detect abnormal texture, we might use a feature that is known to work well in texture classification such as wavelet responses [6]. Or to detect motion in otherwise static camera images, we might calculate the difference between a frame and the previous frame at each position. The simplest choice is an identity function, i.e. the features are the original pixels in the image.

The reason for using feature descriptors instead of simply the images themselves stems from the fact that PCA is a linear model, and the resulting principal components will be combined linearly to reconstruct each image. The problem is that images are not like typical feature vectors, in the sense that (for example) translating an image by a single pixel will result in an almost identical image to the human eye, but a very different feature vector. Moreover, images with texture may look similar to the human eye, but the feature vectors are hardly comparable. Feature descriptors relieve this problem somewhat, by constructing a feature vector that is (at least locally) translation-independent.

A feature can be used to describe an entire image, a specific location, or portions of an image, depending on the descriptor used. This determines how 'localized' the anomaly detection is. For example, we might calculate a locally windowed greyscale histogram, resulting in as many feature vectors as we have windows for each image in the set. We might want to detect entire images as being anomalous, or we might want to focus on specific regions within the image.

b) PCA Decomposition and Partial Reconstruction: The core of this approach is PCA decomposition and partial reconstruction. The rationale is as follows: Common structure within the image set will account for a large amount of the variance present in the set. By decomposing the feature vectors into principal components and discarding components that represent less occurring variations before performing partial reconstruction, we are using PCA akin to a *trained image smoother*, which keeps common and discards uncommon structure.

This step requires a parameter θ, the number of principal components used for reconstruction. This parameter corresponds roughly to a bias/variance trade-off. A very high θ might result in overfitting on the training set as the difference between original and reconstructed feature vectors might contain mostly noise, and processing an image that the method was not trained on would not work well. A very low θ means the method relies more on deviations from the mean feature vector, en less on the deviations it might learn from the training set. It is also possible to replace this abstract parameter θ with a more interpretable concept by choosing a percentage of explained variance that the model should learn, and setting θ to the lowest number of principal components that explain at least that amount of variance.

Note that when using local feature descriptors, PCA should be trained separately for each location in the images, since we have assumed that the image sets are aligned, and thus local feature vectors should be compared only to other feature vectors extracted at the same location in different images.

c) Dissimilarity Function: To determine whether something is or isn't an outlier, the decomposed and reconstructed

local feature vector is compared to the extracted local feature vector by means of some dissimilarity function. The choice for this function again depends on the type of anomaly that is to be detected, as well as the features chosen in step (i). This might be a sum of squared errors, (one minus) a normalized Pearson correlation, or however the chosen feature descriptors are usually matched in other applications. It should be noted that PCA minimizes the mean squared reconstruction error, so this may not be the best dissimilarity to find outliers with, as it is also minimized for the anomalies we want to detect. It can be any function $D(f_1, f_2)$ that compares two feature vectors f_1 and f_2, with only the restriction that $D(f_1, f_1) = 0$, the dissimilarity of a feature vector to itself is zero.

We call the dissimilarity of the feature vector to its partial reconstruction an *anomaly score*, or anomaly ranking. The anomaly score can then be thresholded to determine whether each feature vector represents an anomalous image or region.

A. Proof of Concept

To illustrate our method, we look at the MNIST reference dataset [7], consisting of 70,000 handwritten digits in greyscale images of dimensions $[28 \times 28]$ (see figure 2 for some examples). We use the identity function as feature descriptor, so that the feature vector is identical to the pixel vector. This means our $\mathbf{X}$ is shaped $[70000 \times 784]$. When we apply PCA to the MNIST dataset, we obtain 784 principal components, which we can reshape into $[28 \times 28]$ images for visual inspection (also known as *eigenimages*), as shown in figure 3 for the first 9 principal components.

Fig. 2. A sample of each digit from the MNIST dataset.

Fig. 3. The mean values (left) and first 9 principal components of the MNIST dataset. (Greyscale ranges have been rescaled for maximum visibility.)

Now when we project an image onto the basis spanned by the principal components, we express the image as a linear combination of the eigenimages. Since the eigenimages are sorted in order of decreasing explained variance in $\mathbf{X}$, an image that is similar to the images in $\mathbf{X}$ (in this case: also a handwritten digit, for example) is expected to have a larger (absolute) projected component onto earlier principal components (higher eigenvalues), than onto later principal components (lower eigenvalues).

This is usually the goal when PCA is employed: we project onto the first 2 or 3 principal components and make a scatterplot, or we use it to reduce the dimensionality by one or more orders of magnitude, while reducing the variance by only a fraction. (To illustrate: 90% of the variance in the MNIST dataset is explained in the first 87 principal components, a little more than 11% of the original dimensionality.)

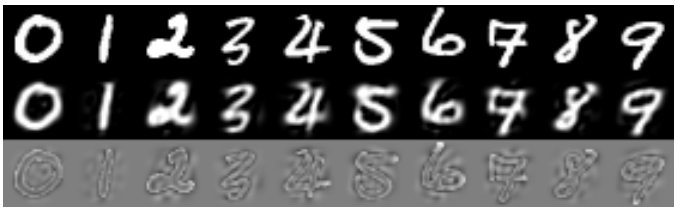


Fig. 4. 10 sample digits from the MNIST dataset (top row) are reconstructed with the first 50 principal components (middle row) and the difference images between the original and the reconstructions (bottom row).

When we project an observation onto all principal components, we can perfectly recreate the original observation by inverting the projection matrix and adding the mean values, but we also know that principal components with lower eigenvalues are expected to be less important, because less of the variance in dataset $\mathbf{X}$ is explained by these components. This leads to the following experiment: an observation is projected onto the first θ principal components, and this projection is augmented with $d - \theta$ zeroes for all less significant principal components we did not project onto. This augmented projection is then projected back. What we get is an approximation of the original observation, as can be seen in figure 4 for $\theta = 50$.

As we can see, the approximations with only 50 principal components are very close to the original images. This is because the PCA was trained on these types of images, and the digits in the example must be similar to the rest of the dataset.

Now what happens when our dataset contains anomalies? To illustrate, we add the first 1,000 images of the CIFAR-10 dataset of natural images [8] to the MNIST dataset[3]. These images are very different from the digits in the MNIST set, and since there are so few of them compared to the total size of the dataset, they can be considered anomalies. We train PCA on the combined dataset and then recreate all images using only the first 50 principal components. We show the reconstruction

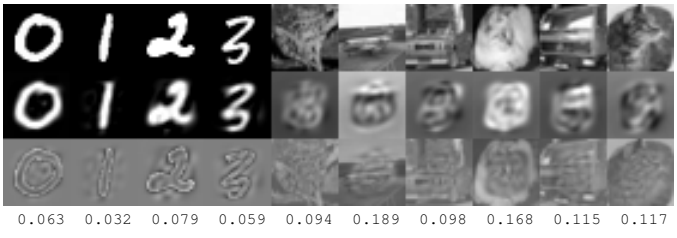


Fig. 5. Sample images from the MNIST and CIFAR-10 datasets (top row) are reconstructed with the first 50 principal components after PCA was trained on 70,000 MNIST images and 1,000 CIFAR-10 images (middle row) and the difference images between the original and the reconstructions (bottom row). Below each difference image are the mean absolute values, which are used as anomaly scores.

[3]The images from CIFAR-10 are converted to greyscale and cropped to $[28 \times 28]$ pixels to conform to the images in the MNIST set.

of some sample images in figure 5. It can be seen here that the images from the CIFAR-10 set reconstruct poorly at the edges, which makes sense as 98.5% of the images are from the MNIST dataset, which does not contain any structure on the edges of the images. As a result, the difference images contain more structure at the edges and the CIFAR-10 images will be easier to distinguish from the MNIST images with our dissimilarity function.

As dissimilarity function, we take the mean absolute value of the pixels in the difference images, which gives us an anomaly score for each image in the set, which is on average going to be higher for images from the CIFAR-10 dataset than images from the MNIST dataset (see for example the anomaly scores of the example images in figure 5). We can now predict which images are anomalies by thresholding the anomaly score.

This illustrates the basic principle of the framework: the reconstruction error with a limited number of principal components can find anomalies in an image set of otherwise similar appearance. Although no feature descriptors were used for this simple example, the need for this will become clear in the next chapter.

IV. Application in Sewer Pipe Images

Dutch sewer engineering company vandervalk+degroot has provided us with a dataset of images from a front-facing camera on a PIG (pipe inspection gadget), from ten different streets within different municipalities in the Netherlands. These images are already spatially aligned, as the inspector has aligned the camera to the center of the pipe before starting. The images contain no labels or annotations though, so a method of verifying that the unsupervised method correctly finds anomalies is required. To this end, we selected two different subsets that are somewhat representative of the all the sewer pipes from the different municipalities present in the datasets and hand-labeled 22 images from these sets.

The two subsets correspond to two different types of pipe: (1) smooth concrete and (2) more rough and textured agglomerate. Figure 6 shows an example of both. Henceforth, we will refer to these two image sets as ‘smooth’ and ‘coarse’.

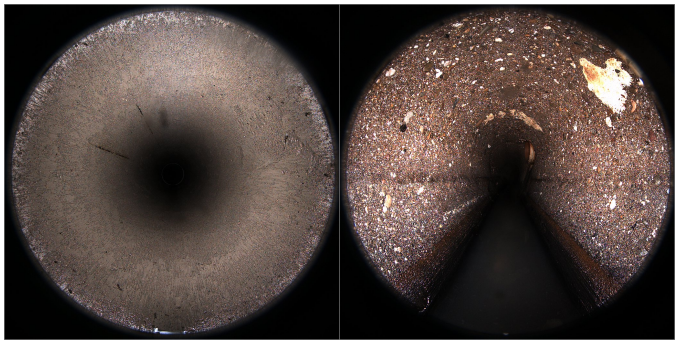


Fig. 6. Sample images from the two labeled datasets: on the left the more smooth concrete pipe, on the right the more roughly textured agglomerate.

The images are processed by the framework on a per-street basis. The reason for this is that the material used varies for different municipalities and date of installation, as will the effects of age. When using images from a single street, we can be reasonably certain that all images in such a set are of similar appearance, which means that anomalies are more easily detected, because we do not have to account for a possible multimodal distribution in appearance.

The images were divided into 324 patches of $[40 \times 40]$ pixels in the regions of the image that are in the focused portion of the images. Each patch in the 22 validation images was labeled as ‘anomaly’ or ‘normal’, in the context of the rest of the pipe. This includes both actual defects, such as discoloration as a result of leakage, as well as physical features that are simply less common than others, such as pipe joints and refuse.

All the images in the sets from which the labeled images originate are divided into the same 324 patches as the labeled images, and for each patch location features are extracted and PCA is applied to the feature vectors at a specific location. This means the framework is applied 324 times and each patch location across the images is treated as a separate image set. We construct an ROC curve by thresholding the anomaly scores at various levels and obtaining true and false positive rates for our labeled validation images. We report the area under the ROC curve (AUROC) as a measure of how well the resulting anomaly score performs.

Note that the parameter θ, the cutoff value for the number of principal components to use in reconstruction, was chosen to maximize the area under the AUROC, but only a single value of θ was used to construct the ROC curve, as opposed to maximizing the curve for every possible value of θ. In our experiments, we found that the optimal value for θ corresponds to approximately 99% explained variance for the smooth image set and 95% explained variance for the coarse image set.

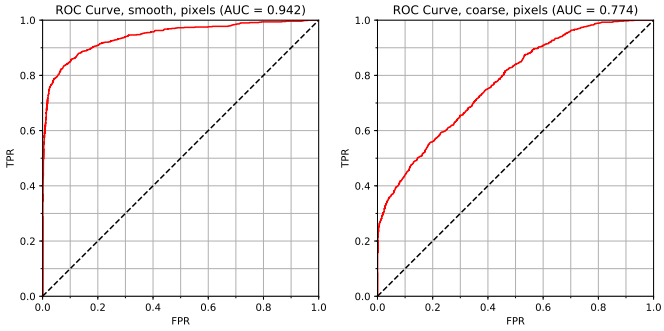


Fig. 7. ROC curves we obtain from the anomaly detection framework on our manually labeled validation set, using pixels as features to be analyzed by PCA. On the left the smooth dataset, on the right the coarse dataset.

When using pixels as features and the mean absolute difference as a dissimilarity measure, we obtain results as shown in figure 7. The AUROC for the smooth set is 0.942, high enough to be of use, however, the AUROC for the coarse set, 0.774, is quite low. This means that if this method were to

see use, human operators would have to sift through a large amount of false positives if the system were calibrated to an acceptable true positive rate.

The reason the framework performs less well on the coarse set when using pixels as features, is the texture present in the surface of the pipe in those images. The variance between pixel values is far greater than it is in the smooth set, where the entire pipe is more or less a single color, and as a result the image are difficult to capture in a linear model such as PCA.

A. Feature Extraction

To alleviate this issue, we extract features that we hope are more robust to textured images. The feature vectors are then decomposed, reconstructed and compared in the same way that the images would be, as shown in the framework in figure 1. In this section, we will propose five higher-level features.

1) Color Histograms: To illustrate the effectiveness, we choose a simplistic feature: a histogram of the pixel values. The 1600 values in each color channel a patch are histogrammed into 20 equally sized bins and concatenated to a feature vector of length 60. These (in comparison) small vectors are decomposed into principal components and reconstructed with fewer than 60 principal components. The histogram is compared to the reconstructed histogram again by mean absolute difference. We see a slight improvement when using the histograms on the coarse set, an AUROC of 0.790, whereas performance on the smooth set is slightly worse with an AUROC of 0.942.

2) Fourier Transform: We perform a 2-dimensional Fourier transform on the $[40 \times 40]$ image patches, obtaining the frequency representation of the image patches. We discard the phase component by taking the absolute value and discard half the frequency plane because of symmetry. Again we use decompose and try to reconstruct the feature vector, using the mean absolute difference as dissimilarity measure. The Fourier transform does not provide an improvement over using the pixel values, as we obtain an AUROC of 0.928 on the smooth set and 0.715 on the coarse set.

3) Histogram of Oriented Gradients: Often abbreviated as HOG, histograms of oriented gradients [9] describe an image determining gradient directions at each pixel location, and binning these locally into histograms over a patch of specified size. It is often used for object recognition. It seems that this feature does not suit our purpose too well, as the AUROC for the smooth set becomes 0.886 and for the coarse set becomes 0.588. This can be explained by the fact that this feature is meant for object detection, and our images contain mostly texture.

4) Local Binary Patterns: Local binary patterns are a feature used to describe points as being edges or corners. [10] Each pixel is compared to its neighboring n pixels (usually $n = 8$) and for each of these neighbors, it assign a 1 or 0 depending on whether the pixel has a higher value than that particular neighbor. The resulting 8-bit numbers are locally histogrammed to summarize the texture of a cell as containing corners, edges, or otherwise. The concatenated histograms are used as a feature vector. We obtain AUROCs of 0.865 for the smooth set and 0.705 for the coarse set.

5) Homogeneous Texture Descriptor: Part of the MPEG-7 multimedia description standard, homogeneous texture descriptors are shown to perform well on image retrieval tasks, even for images with much texture [11]. Simply put, the HTD features are comprised of (logarithmically scaled) mean values and standard deviations of Gabor wavelet responses.

B. Concatenating Feature Vectors

One of the strengths of the framework is that we can concatenate multiple feature vectors and the PCA reconstruction will still function identically. This allows us to combine the strengths of multiple feature types, and even combine these with the raw pixel values if we wish to do so.

Without going into the detail of every possible combination here, we found that excluding the HOG and Fourier transform from the feature vector gave the best result. Figure 8 shows the resulting ROC curves when we use the other high-level features described in this section, as well as the raw pixel values, giving us the highest AUROCs so far, 0.950 for the smooth set and 0.818 for the coarse set.

Leaving out the HOG features seems reasonable, as these performed worse than most other features individually. As both PCA and the Fourier transform are linear operations, performing PCA on the Fourier transform would provide identical results to performing PCA on the pixels (up to an arbitrary phase shift). We discarded the phase component of the Fourier transform before performing PCA, so the result is not identical, but this might explain why including it does not improve the AUROC.

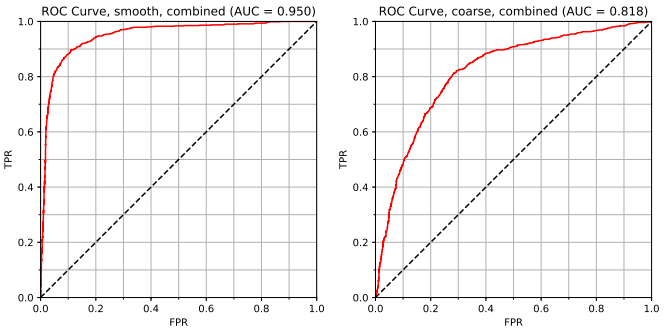


Fig. 8. ROC curves from the anomaly detection framework on the validation set, using both pixel values and the high-level features described in this section (except for HOG and Fourier transform) combined as features to be analyzed by PCA.

V. Convolutional Autoencoder

An autoencoder is a neural network that tries to learn the identity function [12], and a convolutional autoencoder combines this with image filter learning. Analogous to our

framework, this means we can learn the feature representation, perform non-linear dimensionality reduction (replacing the PCA) and reconstruct the input images. As we train this network on an image set, we should be similarly able to use it to detect anomalous regions by inspecting the difference image.

We designed a convolutional autoencoder consisting of:

- Input layer: $[1040 \times 1040]$ resolution
- Convolutional layer 1: 10 $[20 \times 20]$ filters, stride $[10 \times 10]$
- Pooling layer 1: $[2 \times 2]$ max pooling, stride $[2 \times 2]$
- Convolutional layer 2: 10 $[20 \times 20]$ filters, stride $[10 \times 10]$
- Pooling layer 2: $[2 \times 2]$ max pooling, stride $[2 \times 2]$
- Autoencoder: $1690 \rightarrow 845 \rightarrow 422 \rightarrow 845 \rightarrow 1690$ units
- Unpooling layer 1: uniform, $[2 \times 2]$
- Deconvolutional layer 1: Weights shared Conv. layer 2
- Unpooling layer 2: uniform, $[2 \times 2]$
- Deconvolutional layer 2: Weights shared Conv. layer 1
- Output layer: $[1040 \times 1040]$ resolution

Using this network, trained on the same image sets, we obtained the following results: an AUROC of 0.946 on the smooth set and 0.714 on the coarse set, figure 9 shows the ROC curves. The results on the smooth set are rather similar to those obtained by the PCA framework, the results on the coarse set are noticeably worse.

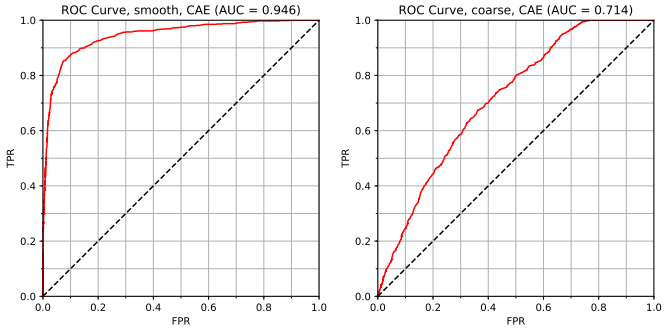


Fig. 9. ROC curves from convolutional autoencoder.

We expect that the reason for this reduced performance is the reconstruction of the full images. In the PCA framework, we are extracting features, decomposing and reconstruction these features, and comparing the reconstruction to the *extracted features*. In the convolutional autoencoder, we try to reconstruct the image itself out of necessity, as we do not know what the features should be. But this means that the reconstructed images are compared to the original images, instead of the reconstructed features to the original features.

To make the systems more similar, we could try to discard the unpooling and deconvolutional layers, and compare the output of the fully connected autoencoder to the input of the fully connected autoencoder (after the network was trained *with* the unpooling and deconvolutional layers), but this is beyond the scope of our current research.

It should also be noted that the metaparameters of the network are far more difficult to optimize than the parameter θ our framework relies on, and a network better designed for this specific task may perform better.

VI. Summary

We have proposed a framework for unsupervised anomaly detection in aligned image sets. Table 1 summarizes the results obtained by the different variants. We see that while raw pixel values perform quite well on the 'smooth' dataset, improvement can be made by combining different feature descriptors. For the 'coarse' dataset, the difference is larger, drastic improvements are made by combining features.

Table 1. Results for the methods and datasets described in this work.

	AUROC	
Feature type	smooth	coarse
Pixels	0.942	0.774
Color Histogram	0.942	0.790
Fourier Transform	0.928	0.715
Histogram of Oriented Gradients	0.886	0.588
Local Binary Patterns	0.865	0.705
Homogeneous Texture Descriptor	0.941	0.785
Pixels + Histogram + LBP + HTD Combined	**0.950**	**0.818**
Convolutional Autoencoder	0.946	0.714

We conclude that our PCA-based approach, which could be considered a more 'traditional' statistical approach to computer vision using combinations of hand-crafted features, outperforms the more 'modern' convolutional autoencoder, but we must also admit that the comparison is not entirely fair as we are in one case reconstructing high-level features and in the other case pixel values.

References

[1] J. Dirksen, F. Clemens *et al.*, "The consistency of visual sewer inspection data," *Structure and Infrastructure Engineering*, vol. 9, no. 3, pp. 214–228, 2013.

[2] TISCA programme funded by NWO-TTW, "Sewersense – multi-sensor condition assessment for sewer asset management," 2016-2020.

[3] K. Pearson, "LIII. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[4] M. David, "Tax. one-class classification; concept-learning in the absence of counter-examples," *ASCI dissertation series*, vol. 65, 2001.

[5] G. O. Campos, A. Zimek *et al.*, "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 891–927, 2016.

[6] M. Unser, "Texture classification and segmentation using wavelet frames," *IEEE Transactions on image processing*, vol. 4, no. 11, pp. 1549–1560, 1995.

[7] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST database of handwritten digits," 1998.

[8] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009, https://www.cs.toronto.edu/~kriz/cifar.html.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[10] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[11] Y. M. Ro, M. Kim *et al.*, "Mpeg-7 homogeneous texture descriptor," *ETRI journal*, vol. 23, no. 2, pp. 41–51, 2001.

[12] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural networks*, vol. 2, no. 1, pp. 53–58, 1989.