Machine learning and computer vision for urban drainage inspections

Meijer, D.W.J.

3 Image-Based Unsupervised Anomaly Detection

In this chapter, 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 emphasise that while this framework 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.

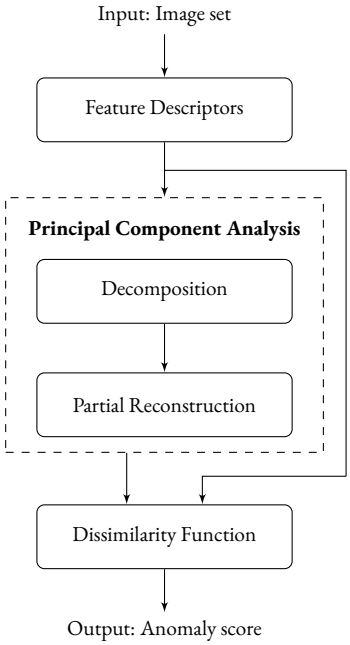

Figure 3.1: The proposed three-part anomaly detection framework.

3.1 Framework

We propose a simple three-part framework to detect local anomalies in aligned image sets and videos, as shown in figure 3.1 and described in more detail in algorithm 1. The three parts are: (i) feature descriptors, (ii) PCA decomposition and partial reconstruction, (iii) a dissimilarity function to compare the PCA reconstructed feature to the extracted features.

Algorithm 1: Anomaly Detection Framework

Input : Image set $\{\mathbf{I}_1, \ldots, \mathbf{I}_N\}$
Input : Feature descriptors $\mathbf{F} : \mathbf{I} \mapsto \mathbb{R}^d$
Input : Number of principal components to use in reconstruction: θ
Input : Dissimilarity function $\mathbf{D} : \left\{\mathbb{R}^d, \mathbb{R}^d\right\} \mapsto \mathbb{R}$
Initialise: Featurespace $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with $\mathbf{x}_i \in \mathbb{R}^d \quad \forall i \in [1, N]$
Initialise: Featurespace $\mathbf{P} = \left\{\mathbf{p}_1, \ldots, \mathbf{p}_N\right\}$ with $\mathbf{p}_i \in \mathbb{R}^d \quad \forall i \in [1, N]$
Initialise: Featurespace $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_N\}$ with $\hat{\mathbf{x}}_i \in \mathbb{R}^d \quad \forall i \in [1, N]$
1 $\mathbf{x}_i \leftarrow \mathbf{F}(\mathbf{I}_i) \quad \forall i \in [1, N]$ `// Extract per-image features`
2 $\mathbf{P} \leftarrow \mathrm{PCA}(\mathbf{X})$ `// Decompose X into PCs`
3 $[\mathbf{p}_i]_j \leftarrow 0 \quad \forall i \in [1, N] \quad \forall j \in (\theta, d]$ `// Discard low variance PCs`
4 $\hat{\mathbf{X}} \leftarrow \mathrm{PCA}^{-1}(\mathbf{P})$ `// Reconstruct to orig. space`
5 $A_i \leftarrow \mathbf{D}\left(\mathbf{x}_i, \hat{\mathbf{x}}_i\right) \quad \forall i \in [1, N]$ `// Calculate anomaly scores`
Output : Anomaly scores $\{A_1, \ldots, A_n\}$

3.1.1 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 common 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 mean the difference between original and reconstructed feature vectors mostly constitutes noise. A very low θ means the method relies more on low-order deviations

from the mean feature vector, and less on the specific deviations it might learn from the entire 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, or even a specific fraction of d.

3.1.2 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 [1]. Or to detect motion in otherwise static camera images, we might calculate the difference between a frame and the previous frame at each position and use these as features. The simplest choice is an identity function, i.e. the features are the original pixel values in the image.

[1] UNSER, M. 1995. Texture classification and segmentation using wavelet frames. *IEEE Transactions on image processing 4*, 11, 1549–1560

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 pixel values are hardly comparable. Extracted features, unlike the images they were extracted from, may have invariances to transformations that makes them more suited to compare images of a certain type than the original pixel values would.

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 'localised' 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. When using localised features, we have the option to either treat all resulting feature vectors as if they came from the different images (treating each window location as an image in itself) or perform the framework for each window location individually.

3.1.3 Dissimilarity Function

To determine whether something is or isn't an outlier, the decomposed and reconstructed feature vector is compared to the feature vector before decomposition by means of some dissimilarity function. This might be Euclidean distance, one minus a normalised Pearson correlation, or however the chosen feature descriptors are usually matched in other applications [2]. It can be any function $D(f_1, f_2)$ that compares two feature vectors f_1 and f_2, with the restrictions that $D(f, f) = 0$, the dissimilarity of any vector to itself is zero, $D \geq 0$ for all inputs, and the function is symmetric: $D(f_1, f_2) = D(f_2, f_1)$. Triangle inequality, $D(f_1, f_3) \leq D(f_1, f_2) + D(f_2, f_3)$, is a property that we might want a dissimilarity function to have, but is not required. If a dissimilarity function does satisfy triangle inequality, it may also be called a *distance function* or *metric*.

[2] It should be noted that PCA minimises the mean squared reconstruction error, so this is also minimised for the anomalies we want to detect.

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

Because the optimal value for thresholding will vary depending on feature descriptor, dissimilarity function and number of principal components used to reconstruct, we will evaluate an AUROC of a manually labeled test set to

Figure 3.2: A sample of each digit from the MNIST dataset.

Figure 3.3: The mean values (left) and first 9 principal components of the MNIST dataset. (Greyscale ranges have been rescaled for maximum visibility.)

assess the quality of this method. The ROC curve itself is also a useful chart to have, as in the case of defect detection for industrial processes (such as sewer inspections) we often have a higher tolerance for false positives than we do for false negatives.

3.2 Proof of Concept

To illustrate our method, we look at the MNIST reference dataset [3], consisting of 70,000 handwritten digits in greyscale images of dimensions $[28 \times 28]$, see figure 3.2 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 feature matrix 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 3.3 for the first 9 principal components.

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

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, an image that is similar to the images in the set (in this case also a handwritten digit, for example) is expected to have a larger (absolute) projected component (or eigenvalue) onto earlier

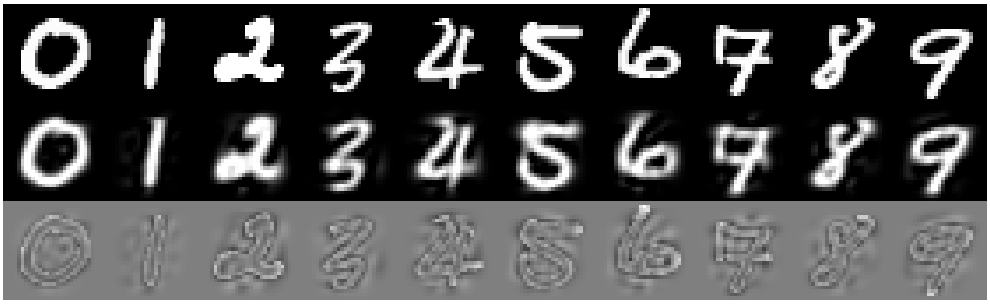

Figure 3.4: 10 sample digits from the MNIST dataset (top row) are reconstructed with the first 50 principal components (middle row) and the difference images between the original and the reconstructions (bottom row).

principal components, than onto later principal components.

The goal when PCA is employed is often dimensionality reduction: we project onto the first 2 or 3 principal components for inspection, 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 in the first 87 principal components, a dimensionality reduction of about 89%.)

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 present in the dataset 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 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 3.4 for the MNIST dataset and $\theta = 50$ (a dimensionality reduction of over 95%).

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

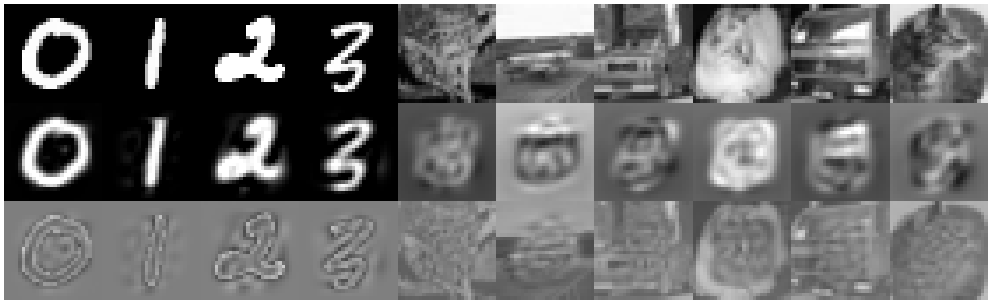

Figure 3.5: Sample images from the MNIST and CIFAR-10 datasets (top row) are reconstructed with the first 50 principal components after PCA was performed on 70,000 MNIST images and 1,000 CIFAR-10 images (middle row) and the difference images between the original and the reconstructions (bottom row). Below each difference image is the mean absolute value, which is used as the anomaly score.

[4] KRIZHEVSKY, A. AND HINTON, G. 2009. Learning multiple layers of features from tiny images. `https://www.cs.toronto.edu/~kriz/cifar.html`

[5] 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.

the digits in the example are 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 [4] to the MNIST dataset [5]. 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 perform PCA on the combined dataset and then recreate all images using only the first 50 principal components. We show the reconstruction of some sample images in figure 3.5.

It can be seen 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. This is going to be categorically 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 3.5). We can now predict which images are anomalies by thresholding

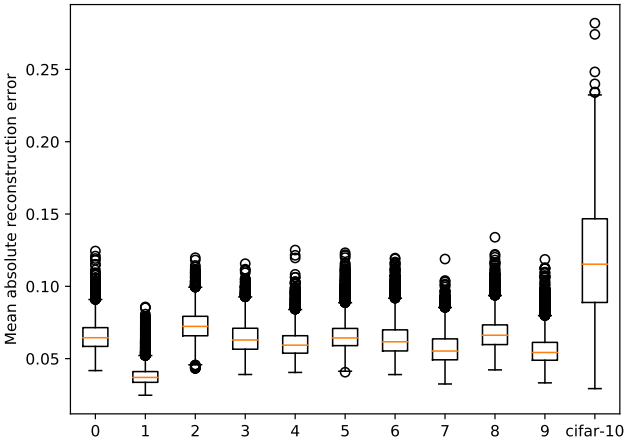

Figure 3.6: Comparison of the absolute reconstruction error of the 70,000 digits in the MNIST dataset and the first 1,000 images of the CIFAR-10 dataset, using the first 50 principal components to recreate the images.

the anomaly score. Figure 3.6 shows the spread of the reconstruction errors for different digits and images from the CIFAR-10 set. As can be seen, the error of the CIFAR-10 images tends to be significantly larger.

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 section.

3.3 Application in Sewer Pipe Images

Dutch urban drainage inspection 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

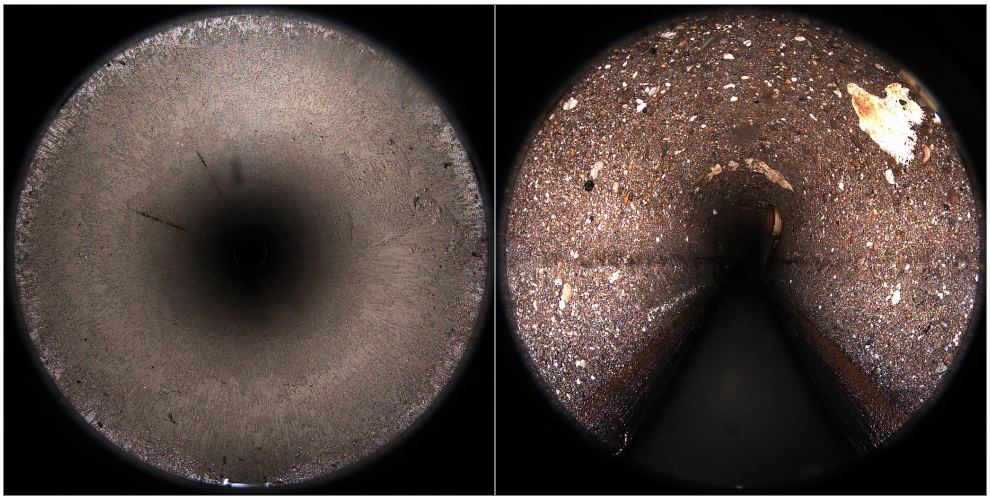

Figure 3.7: Sample images from the two labeled datasets: on the left the more smooth concrete pipe, on the right the more roughly textured granulate.

has aligned the camera to the centre of the pipe before starting the recording.

The two subsets correspond to two different types of pipe: (1) smooth concrete and (2) more rough and textured granulate, exposed over time. Figure 3.7 shows an example of each. Henceforth, we will refer to these two image sets as 'smooth' and 'coarse'. The image sets contain 684 and 698 images respectively. Each individual image is composed of $[1080 \times 1080]$ RGB pixels.

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 manufacturing and age, which means that anomalies are more easily detected, because we do not have to account for a possible multimodal distribution in appearance.

The images are divided into 676 non-overlapping patches of $[40 \times 40]$ pixels, and we select 324 of such patches per image, corresponding to the regiosn of the images that are

in focus. Different patch locations are processed separately, this allows us to compare the portions of the images that are spatially aligned while high anomaly scores can still be pinpointed to a specific image patch, rather than an entire image.

The images are not accompanied by labels or annotations, 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 all the sewer pipes from the different municipalities present in the datasets and hand-labeled 22 images from these sets. 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 discolouration as a result of leakage, as well as physical features that are simply less common than others, such as pipe joints and refuse.

The images in the labeled subsets sets 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.

The parameter θ, the cutoff value for the number of principal components to use in reconstruction, was chosen to maximise the AUROC. 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.

The AUROC allows us to compare the performance of different methods regardless of what costs or restrictions we

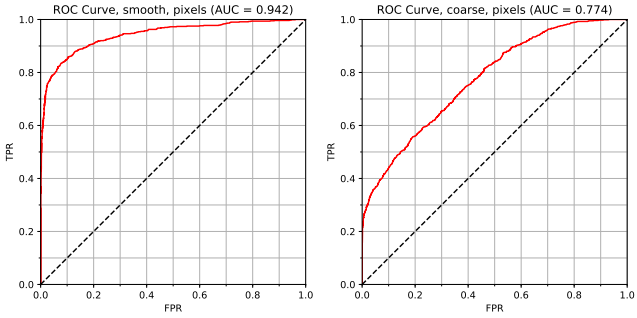

Figure 3.8: ROC curves we obtain from the anomaly detection framework on our manually labeled validation set, using pixels as features to be analysed by PCA. On the left the smooth dataset, on the right the coarse dataset.

assign to types of misclassification. It should also be noted that since this anomaly detection step might be followed by a classification into a taxonomy of defect classes, a high false positive rate might be salvaged by the later classification.

When using pixels as features and the mean absolute difference as a dissimilarity measure, we obtain results as shown in figure 3.8. The AUROC for the smooth set is 0.942, the AUROC for the coarse set is 0.774.

3.3.1 Feature Extraction

A possible reason that 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 colour, and as a result the image are difficult to capture in a linear model such as PCA.

To alleviate this issue, we extract features that 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 3.1. In this section, we propose five higher-level features. An overview of each feature's invariances is given in table 3.1. The performances of each can be easily compared

Feature	Invariances
Pixel Values	None
Colour Histogram	Translation, rotation, scaling
Fourier Transform	Translation [6]
Histogram of Oriented Gradients	None
Local Binary Patterns	Translation, rotation
Homogeneous Texture Descriptor	Translation, rotation

Table 3.1: Overview of feature extractors invariances

[6] After discarding phase component

in table 3.2.

Colour Histograms

A simplistic but quite useful feature is a colour histogram of the pixel values. The 1600 values in each colour channel of a patch are binned into 20 equally sized bins per colour channel and concatenated to form 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 similar with an AUROC of 0.942.

Fourier Transform

We perform a two-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 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.

Histogram of Oriented Gradients

Often abbreviated as HOG, histograms of oriented gradi-

ents [7] describe an image by determining gradient directions at each pixel location, and binning these locally into histograms over a patch of specified size. 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 might be explained by the fact that this feature is meant for object detection, and our image patches contain mostly texture.

[7] DALAL, N. AND TRIGGS, B. 2005. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE, 886–893

LOCAL BINARY PATTERNS

Local binary patterns are a feature used to describe points as being edges or corners [8]. Each pixel is compared to its neighbouring n pixels (usually $n = 8$) and for each of these neighbours, it assigns a 1 or 0 depending on whether the pixel has a higher greyscale value than that particular neighbour. The resulting 8-bit numbers are locally binned to summarise 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.

[8] OJALA, T., PIETIKÄINEN, M., AND HARWOOD, D. 1996. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition 29,* 1, 51–59

HOMOGENEOUS TEXTURE DESCRIPTOR

Part of the MPEG-7 multimedia description standard, homogeneous texture descriptors are shown to perform well on image retrieval tasks, especially for images with much texture [9]. The HTD features are comprised of logarithmically scaled mean values and standard deviations of Gabor wavelet responses. We obtain AUROCs of 0.941 for the smooth set and 0.785 for the coarse set.

[9] RO, Y. M., KIM, M., KANG, H. K., MANJUNATH, B., AND KIM, J. 2001. MPEG-7 homogeneous texture descriptor. *ETRI journal 23,* 2, 41–51

3.3.2 CONCATENATING FEATURE VECTORS

One of the strengths of the framework is that we can concatenate multiple feature vectors and the framework will

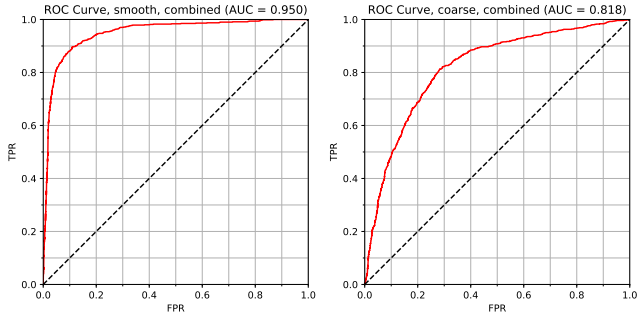

Figure 3.9: ROC curves from the anomaly detection framework on the validation set, using both pixel values and the high-level features described in this section (except for HOG and Fourier transform) combined as features to be analysed by PCA.

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.

After examining every possible permutation of the features previously described, we found that excluding only the HOG and Fourier transform from the feature vector gave the best result on both image sets. Figure 3.9 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. The ROC curves are shown in figure 3.9.

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 (excluding 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 when already using the pixel values does not improve the AUROC.

3.4 Convolutional Autoencoder

Principal component analysis is not the only available method for this task. We compare the performance of this method when using a convolutional autoencoder as a drop-in replacement.

An autoencoder is a neural network that tries to learn the identity function [10], and a convolutional autoencoder combines this with image filter learning [11]. 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 × 1040] resolution
- Convolutional layer 1: 10 [20 × 20] filters, stride [10 × 10]
- Pooling layer 1: [2 × 2] max pooling, stride [2 × 2]
- Convolutional layer 2: 10 [20 × 20] filters, stride [10 × 10]
- Pooling layer 2: [2 × 2] max pooling, stride [2 × 2]
- Autoencoder: 1690 → 845 → 422 → 845 → 1690 units
- Unpooling layer 1: uniform, [2 × 2]
- Deconvolutional layer 1: Weights shared Conv. layer 2
- Unpooling layer 2: uniform, [2 × 2]
- Deconvolutional layer 2: Weights shared Conv. layer 1
- Output layer: [1040 × 1040] resolution

[10] Baldi, P. and Hornik, K. 1989. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks 2*, 1, 53–58

[11] Chen, M., Shi, X., Zhang, Y., Wu, D., and Guizani, M. 2017. Deep features learning for medical image analysis with convolutional autoencoder neural network. *IEEE Transactions on Big Data*

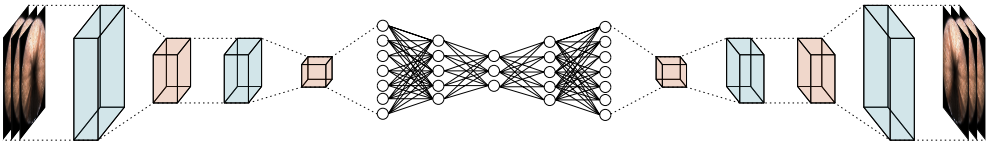

Figure 3.10: Schematic overview of the convolutional autoencoder used. Convolutional and deconvolutional layers are shown in blue, pooling and unpooling layers are shown in orange.

A schematic overview of the network can be found in figure 3.10.

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 3.11 shows the ROC curves. The results on the smooth set are rather similar to those obtained by the PCA framework, the AUROC results on the coarse set are noticeably worse, as can be seen when comparing with the PCA-based method in table 3.2.

Still, urban drainage inspections might be an application where the convolutional autoencoder could outperform the PCA-based method, when we cannot afford to miss any potential defects. We can see from comparing the ROC curves that the convolutional autoencoder reaches a true positive rate of 1.0 at a lower false positive rate than the PCA-based method. Overall performance is still expected to be worse, as indicated by the AUROC.

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.

The fact that the convolutional autoencoder has to reconstruct the original image, means it can't learn features we might describe as 'texture descriptors,' as these are inher-

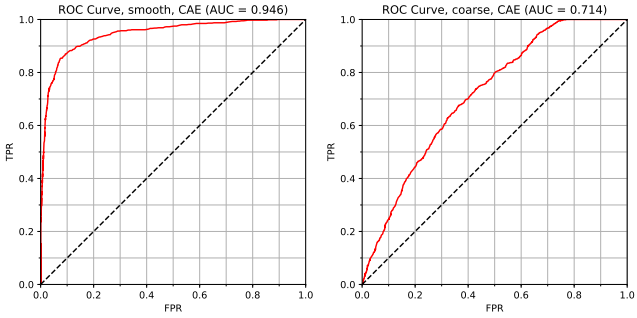

Figure 3.11: ROC curves from convolutional autoencoder.

Table 3.2: Results for the methods and datasets described in this work.

	AUROC	
Feature type	smooth	coarse
Pixels	0.942	0.774
Colour 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	**0.950**	**0.818**
Convolutional Autoencoder	0.946	0.714

ently rotation and translation independent, so reconstructing the original pixel values from such features would be impossible for patches containing a lot of texture. But these are the types of features we expect (and confirmed for the PCA-based approach) to perform well, so the comparison is not entirely fair.

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 network's many hyperparameters are more difficult to optimise than the singular parameter θ our framework relies on, and a network better optimised for this specific task may perform better [12].

[12] SUN, Y., XUE, B., ZHANG, M., AND YEN, G. G. 2018. An experimental study on hyper-parameter optimization for stacked auto-encoders. In *2018 IEEE Congress on Evolutionary Computation (CEC)*. 1–8

3.5 Summary

We have proposed a framework for unsupervised anomaly detection in aligned image sets, relying on feature extraction, PCA decomposition and partial reconstruction, and classification of the reconstruction error, and tested this framework on sewer pipe images. Table 3.2 summarises the results obtained by the different feature types. 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, as is expected when we consider that the images are more defined by texture than by individual pixel values.

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 in this setting, 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.