HOCHSCHULE MITTWEIDA
University of Applied Sciences

Applied Computer Sciences and Biosciences

Bachelor Thesis

AI based Anomaly Detection for Banknotes

Author:

Kornelije Juric

Study Programme:
Applied Mathematics B.Sc.

Seminar Group:
MA20w1-B

Supervisors:

Prof. Dr.-Ing. Alexander Lampe

M.Sc. Thomas Davies

April 21, 2024

Abstract

The detection of anomalies is one of the key problems occurring for example in commercial quality control applications. This work explores the potentials of a novel machine learning approach referred to as Reconstruction by Inpainting for Visual Anomaly Detection (RIAD), based on encoder-decoder architecture and image inpainting techniques. The approach is applied to the task of detecting anomalies on banknote images such as stains and scribbles while having to cope with inherent banknote print variations. Using a dataset consisting of €50 banknotes, rigorous experimentation is conducted to evaluate the efficacy of this approach and explore simpler and faster solutions. This study aims to offer practical solutions for automating the assessment of banknote fitness, with potential applications in improving the efficiency of currency processing in ATMs and money-counting machines.

Acknowledgement

Thanks to Prof. Lampe, for giving me the thesis topic, for regularly supervising my research with enthusiasm and for providing insights and inspiration.
Thanks to Thomas for always being ready to help in moments of need.
Thanks to HS-Mittweida and all the staff for this opportunity.
Thanks to my family for all the love and support.
Thanks to every person I met during my studies in Mittweida for enriching my experience.
Thanks _ for leading me so far.

Contents

1 Introduction

Banknotes, as a widely circulated medium of exchange, are susceptible to various forms of degradation, including stains from ink marks, scribbles, environmental factors, and handling. These stains not only affect the aesthetics of banknotes but may also raise concerns regarding their authenticity and usability.

Throughout history, banknotes have also served as a potent tool for propaganda. There are plenty of instances where warring factions have employed banknotes to disseminate ideological messages and strengthen morale [1]. However, beyond wartime contexts, certain organizations have utilized banknotes as a means to propagate messages that challenge or undermine the authority of central governments, or to influence political sentiments. As an example, the Falun Gong movement in China [2] [3], as shown in Figure 1. Such use of currency underscores its potential as a vehicle for shaping public opinion and exerting social and political influence.

Figure 1:
Falun Gong movement message
on Chinese banknote. Translation:

"How many prophets have warned,
humanity knows great decay,
retreat from the ranks and
levels of the Chinese Communist Party,
and wait for the moment till
the Great Law will guard peace" [3].

Anomaly detection would be an easy task if all banknotes of the same kind looked the same, a pixel-wise difference would solve the problem, but many details make it a challenging task. When banknotes are printed, they start from big sheets of paper on which layers are printed one after another. These layers have some positional tolerance that's easily visible when scrolling through banknote images. What I'm referring to can be observed in this video [4]. Then we have some security elements, that are light-reflecting, and depending on the microsecond in which the scanner has captured the image, unique patterns can emerge. The two serial numbers on the back side are also a unique combination of numbers and letters, as shown in Figure 2. On the front upper left corner, under the EU flag, there might be a signature either from Christine Lagarde or from Mario Draghi. And lastly, in the same corner, the year is changing.

This study focuses mainly on real banknotes that have been altered in circulation. Counterfeit detection is beyond the scope of this research.

We are aware that this is an area of research for companies specialising in the production of sensors used in Automated Teller Machines (ATMs) or Automated Sorting Machines. While the existing systems may incorporate some form of automated authentication and counterfeit detection, the specific implementation of machine-learning algorithms for stain and scribble detection may represent a novel contribution with the potential of enhanced accuracy and efficiency in currency processing. The speed of execution is especially a challenge in such devices with limited computational power.

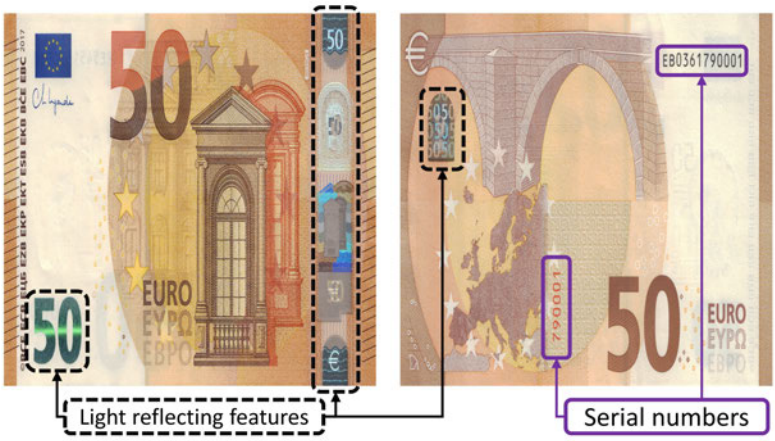


Figure 2: 50 Euro banknote features.

This thesis explores in depth the RIAD approach, presented in the paper "Reconstruction by inpainting for visual anomaly detection" [5], which demonstrated very good performance on most objects from the MVTec AD [6] dataset for industrial inspection. Instead of the MVTec AD dataset, our own data set consisting of €50 banknotes was created and used for the experiments. The approach uses a variation of a U-Net [7] combined with image inpainting techniques. Extensive experiments were conducted and weak points of the method were detected, specifically with the MSGMS similarity measure. An alternative similarity measure based on HSV color space is presented and results based on both similarity measures are closely compared. Further experiments and ablation studies are performed to reduce the time complexity of the method.

2 Preliminaries

2.1 Color Spaces

Digital images are composed of pixels, each representing a tiny dot of color. The color of each pixel is typically described using a combination of color channels. Two common representations of color in digital images are the RGB (Red, Green, Blue) and HSV (Hue, Saturation, Value) color spaces [8].

RGB: In the RGB color space, each pixel is represented by three color channels: red, green, and blue. The intensity of each channel determines the amount of red, green, and blue light present in the pixel. Typically the pixel values range from 0 to 255 for each channel.

HSV: The HSV color space represents colors in terms of three components:

- **Hue (H)** refers to the dominant wavelength of the color and is represented as an angle around a color wheel. It defines the perceived color, such as red, green, or blue.
- **Saturation (S)** quantifies the purity or intensity of the color. Higher saturation values indicate more vivid colors, while lower values result in more muted shades.
- **Value (V)** represents the brightness or intensity of the color. It determines how light or dark the color appears [8].

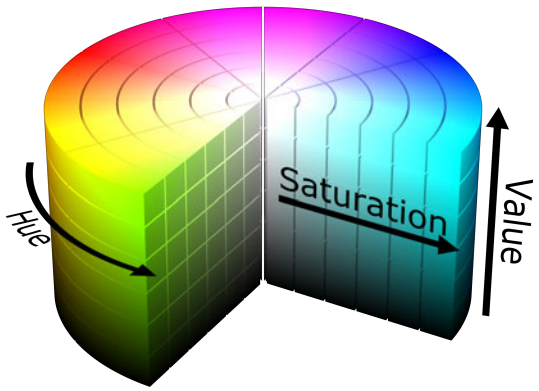


Figure 3: HSV color space. Figure taken from [9]

2.2 Convolutional Operations

Convolutional operations are fundamental building blocks in convolutional neural networks (CNNs), particularly used in computer vision tasks. Convolutional layers in CNNs apply convolutional operations to input data, typically images, to extract meaningful features. The convolution operation between an input I and a kernel K is defined as follows:

$$(I * K)(i, j) = \sum_m \sum_n I(m, n) \cdot K(i - m, j - n) \tag{1}$$

where (i, j) represents the spatial location in the output feature map, and the summation is over the spatial dimensions of the input I and the kernel K [10].

Convolutional operations are characterized by several parameters [11]:

The **kernel (filter)** is a small matrix of weights used for the convolution operation. It defines the spatial extent of the operation and the pattern that the convolutional layer aims to detect. Common kernel sizes are 3×3, 5×5, and 7×7, although other sizes are also used depending on the specific application.

The **stride** determines the step size at which the kernel moves across the input image. A stride of 1 means that the kernel moves one pixel at a time. A larger stride reduces the spatial dimensions of the output feature map, leading to more aggressive downsampling.

The **padding** refers to the process of adding additional border pixels to the input image. It is often necessary to preserve the spatial dimensions of the input and output feature maps, especially when using large kernels or strides. Zero-padding (adding zero-value pixels) is the most common padding technique, although other padding methods exist, such as reflection padding and replicate padding.

The **output size** of the feature map depends on the spatial dimensions of the input, the kernel size, the stride, and the padding. Given an input image of size $H \times W$ and a kernel of size $K \times K$, the output size O can be computed using the formula [11]:

$$O = \left\lfloor \frac{H - K + 2 \times \text{padding}}{\text{stride}} \right\rfloor + 1$$

2.3 SSIM

The **Structural Similarity Index Measure (SSIM)** is a widely used metric for assessing the similarity between two images. It was introduced by Wang et al. in their paper [12] in 2004. SSIM is designed to capture the perceptual difference between two images, taking into account both luminance and structural information.

The SSIM index between two images, x and y, is computed as follows [12]:

$$\text{SSIM}(x, y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \cdot \frac{2\sigma_{xy} + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \tag{2}$$

where:

- μ_x and μ_y are the means of x and y, respectively.
- σ_x^2 and σ_y^2 are the variances of x and y, respectively.
- σ_{xy} is the covariance of x and y.
- c_1 and c_2 are constants added to avoid instability when the denominator is close to zero.

The SSIM index ranges from -1 to 1, where 1 indicates perfect similarity, 0 indicates no similarity, and -1 indicates complete dissimilarity [12].

2.4 Performance Metrics

Since this study focuses on the task of deciding whether an image contains an anomaly or not, the task can be formulated as a binary classification task. Some of the common performance metrics used for evaluating classification models are explained in this subsection.

Confusion Matrix is a tabular representation of the model's predictions compared to the actual labels. It provides insights into the model's performance across different classes, including true positives, false positives, true negatives, and false negatives. Evaluation of classification models often involves a trade-off between precision and recall, depending on the specific application requirements. For example, in medical diagnosis tasks, achieving high recall (sensitivity) is typically more important than precision to minimize false negatives [13].

Accuracy measures the proportion of correctly classified data points over the total number of data points. It is computed as the ratio of the number of correctly predicted instances to the total number of instances.

Precision quantifies the proportion of true positive predictions among all positive predictions made by the model. It is computed as the ratio of true positives to the sum of true positives and false positives.

Recall, also known as sensitivity, measures the proportion of true positive predictions among all actual positive instances in the dataset. It is computed as the ratio of true positives to the sum of true positives and false negatives.

F1 Score is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance by considering both precision and recall. It is computed as $2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$.

ROC (Receiver Operating Characteristic) curve is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) for different threshold values.

AUROC (Area Under the Receiver Operating Characteristic) measures the area under the ROC curve, which quantifies the overall performance of the binary classifier. An AUROC value of 1 indicates a perfect classifier, while a value of 0.5 suggests a random classifier [13].

3 RIAD

After extensive research into various autoencoder-based anomaly detection methods, the RIAD approach, presented in the paper "Reconstruction by inpainting for visual anomaly detection" [5], emerged as a reasonable choice for our application case. RIAD offers a promising approach by utilizing inpainting techniques to reconstruct anomalous regions within images. It has demonstrated robustness across different anomaly types and datasets, especially the MVTec AD dataset, showcasing its potential for anomaly detection within industrial inspection. The goal of the thesis will be to examine whether this model will be able to deal with the changing serial numbers and the light-reflecting components of Euro banknotes.

3.1 Masking

Image Inpainting is the task of reconstructing missing or damaged parts of an image. The idea is to teach the model to reconstruct images without an anomaly. So for the training, we only use images of clean banknotes. A portion of the image is randomly masked, and the model then reconstructs the masked regions. In the inference then given an image of a banknote with an anomaly, the anomalous regions hopefully will be masked and the model should reconstruct a clean banknote as it learned to do during the training, as shown in Figure 4

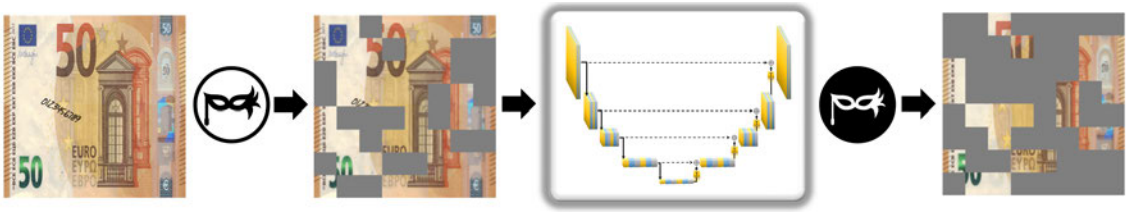


Figure 4: The masking process (Note that the grey patches are due to normalization during tensor to array transformation, but otherwise they should be black since in those areas we are setting all 3 color channels to zero).

The accuracy of inpainting depends on the anomaly size and the size of masked regions. Therefore as in [5] we use several sizes for the masked regions. Let `img_size` be the width = height of our input image. For masking, we use squared patches for simplicity, so let `k` be the width = height of the masked patches, sampled for some list of k values. In our case `img_size = 256` and `k_list = [4, 8, 16, 32]`. For each `k` in `k_list` we want to generate `n` disjoint masks, which together cover the whole image. This way, the anomaly will be masked for sure and in its place a clear part of the banknote will be reconstructed. The n here is a hyperparameter which we set to `n = 3` as in [5].

Let N `= (img_size / k)` be the number of patches along one axis and N^2 the total number of patches for the given `k`. We then randomly permute a sequence of indices in the range $(0, N^2 - 1)$ and further split it into n smaller sequences S_i. These n sequences of indices are then used to construct n masks M_i.

$$M_i(x) = \begin{cases} 0, & \text{if } x \in S_i \\ 1, & \text{otherwise} \end{cases} \qquad \text{for } x \in [1, N^2]$$

Perhaps the following Python code explains it better:

```
1  import numpy as np
2
3  img_size = 256
4  k_list = [4, 8, 16, 32]
5  n = 3
6  mask_list = []
7
8  for k in k_list:
9      N = img_size // k
10     rand_sequence = np.random.permutation(N*N)
11     remainder = (N*N) % n
12     if remainder > 0:
13         rand_sequence = np.concatenate((rand_sequence,
14                                         np.asarray([-1] * (n - remainder))))
15     n_sequences = rand_sequence.reshape(n, -1)
16     for sequence in n_sequences:
17         mask = [0 if i in sequence else 1 for i in range(N*N)]
18         mask = np.asarray(mask).reshape(N, N)
19         mask = mask.repeat(k, 0).repeat(k, 1)
20         mask_list.append(mask)
```

In lines 11-14 the `remainder` deals with the fact that N^2 might not be divisible by n. So we compensate by adding -1s in at the end of the sequence. In lines 17-19 the variable `mask` is first a binary list, then it becomes an NxN matrix and finally a binary matrix that matches the size of our input image. The `mask.repeat()` is "stretching" our mask in both directions, it repeats each element of an array, k times in the specified direction. Take as an example the following:

$$\texttt{A} = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \qquad \texttt{A.repeat(2,0).repeat(2,1)} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 2 & 2 & 3 & 3 \\ 2 & 2 & 3 & 3 \end{bmatrix}$$

At the end, the `mask_list` will contain `n * len(k_list)` binary masks M, in our case 12, each of which is pixel-wise multiplied with our input image I,

$$I_M(i) = I \cdot M_{(i)} \qquad for \quad i \in [1, n]$$

the masked input I_M is then fed through the U-Net, which reconstructs the patched regions. The outputs $I_{r(i)}$ are then masked again with the inverse of the masks used initially and the 3 disjoint subsets are summed up composing the whole reconstruction I_R.

$$I_R = \sum_{i=1}^{n} I_{r(i)} \cdot (1 - M_{(i)})$$

In the end, this yields k reconstructions I_R and for each one of them a similarity measure with respect to the input image is computed and an anomaly map is generated. We will see more on this in the section 3.3. Figure 5 gives an overview of the whole RIAD approach. Next, we will dive deeper into the U-Net architecture.

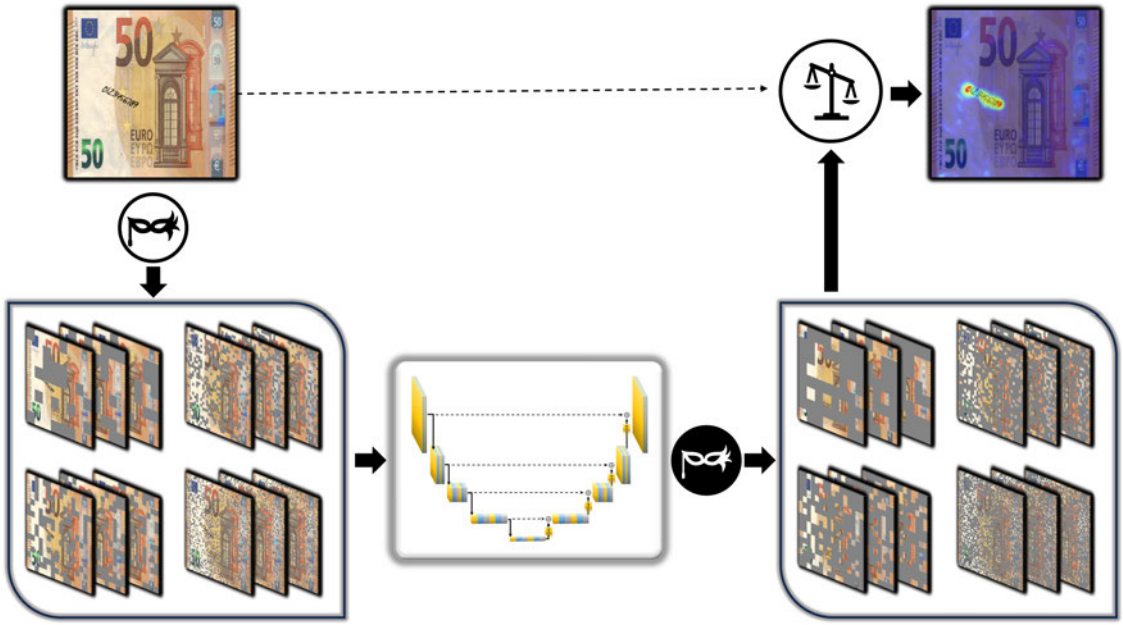


Figure 5: RIAD overview: given an input image, the masking is applied, generating multiple inputs for the pre-trained U-Net whose outputs are then masked again with the inverse of the initial masks. The masked outputs are then summed and weighted against the initial input image with some similarity measure to produce the anomaly map.

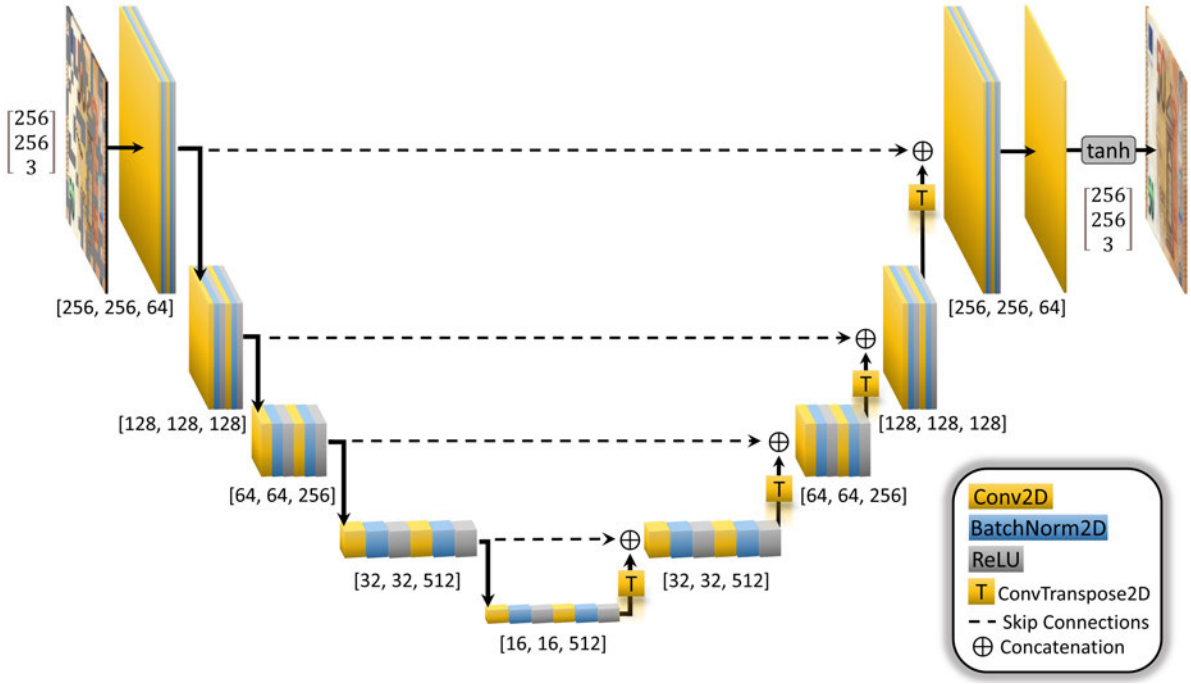


Figure 6: The U-Net architecture used in our study.

3.2 U-Net

The encoder-decoder network based on the U-Net architecture [7], which was originally designed for medical image segmentation tasks, is used to reconstruct the masked parts of the image. The network comprises several down and up blocks, each incorporating a double block of convolutional layer, batch normalization, and rectified linear unit (ReLU) activation functions, as in Figure 6.

The down blocks progressively reduce spatial dimension and increase feature channels by means of convolution, while the up blocks reverse this process. In the up block, the transposed convolution is responsible for upsampling the spatial dimension while the standard convolution decreases the feature channels.

In convolutional operations, we have hyperparameters such as kernel size, stride, and padding. While in the standard convolution, stride determines the step size at which the filter is applied across the input, in convolution transpose we add $z = (stride - 1)$ of zeros between each row and column of the input, and the step size is always 1. Figures 7 and 8 serve as examples.

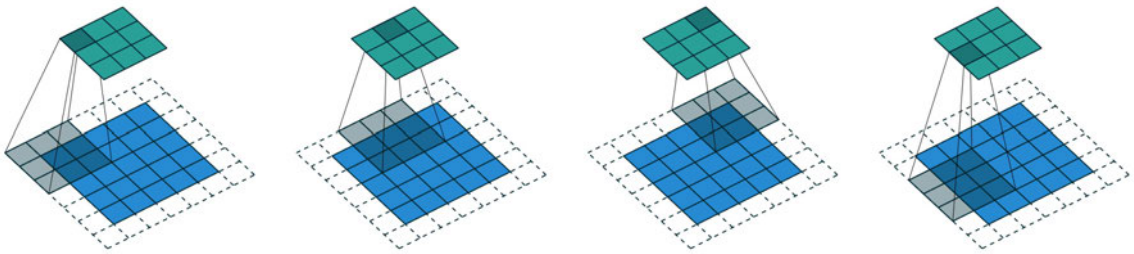


Figure 7: Convolution operation with kernel size = 3 and stride = 2 on a 5×5 input with padding = 1 . Figure taken from [11].

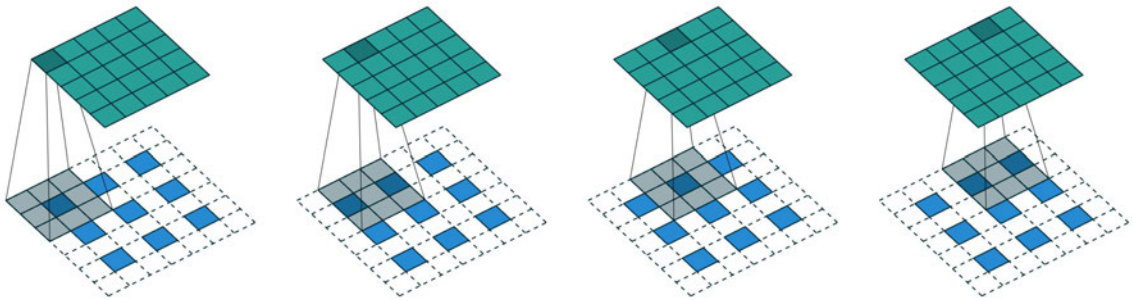


Figure 8: The transpose of the convolution in Figure 7. It is equivalent to convolving a kernel of size = 3 with stride = 1 over a 3×3 input (with zeros inserted in between) with padding = 1. Figure taken from [11]. More examples with animation can be seen here [14].

In our case, in each down block, only the first convolutional layer is responsible for downsampling the input feature maps by a factor of 2 through the application of a convolutional filter with a stride of 2. The second convolutional layer in the down block continues to extract and refine features, capturing more abstract and high-level representations, without an effect on spatial dimension and channel count. A padding of 1 is set to every convolutional layer in the network

Downsampling						
Down Block	Conv 1				Conv 2	
	channels in	channels out	kernel size	stride	kernel size	stride
1	3	64	3	1	3	1
2	64	128	4	2	3	1
3	128	256	4	2	3	1
4	256	512	4	2	3	1
5	512	512	4	2	3	1

In the up blocks, the output from the previous layer undergoes transpose convolution, a process that increases the spatial dimensions while decreasing the number of feature channels by a factor of 2. This operation upsamples the feature maps to match the resolution of the corresponding feature maps from the down blocks.

Skip connections are then employed to transfer the outputs from the corresponding down blocks, facilitating the fusion of low-level and high-level features. By integrating information from multiple scales, skip connections enable the network to reconstruct detailed information while preserving spatial coherence and improving the overall quality of the reconstructed output [5].

These two outputs are concatenated along the channel dimension and passed to the first convolutional layer which progressively halves the number of channels. The second convolutional layer as in the down blocks, does not affect the dimension of the feature maps.

Upsampling										
Up Block	Conv. Transpose				Conv. 1				Conv. 2	
	in	out	kernel	stride	in	out	kernel	stride	kernel	stride
1	512	512	4	2	1024	512	3	1	3	1
2	512	256	4	2	512	256	3	1	3	1
3	256	128	4	2	256	128	3	1	3	1
4	128	64	4	2	128	64	3	1	3	1

Each convolution is followed by a batch normalization and an activation function ReLU. Batch normalization is primarily used to address the problem of internal covariate shift by normalizing the activations of each layer. This normalization process helps stabilize the training process and accelerates convergence, leading to faster and more stable training of deep neural networks. Additionally, it acts as a regularizer by introducing noise into the network acting as a regularizer during the training [15].

The final layer consists of a convolutional layer followed by a hyperbolic tangent (tanh) activation function which outputs our reconstruction.

3.3 Similarity Measures

When training auto-encoders, it's typical to employ a pixel-wise L2 loss, but this approach assumes independence between neighbouring pixels, i.e. changes in one pixel's intensity or color do not affect the intensity or color of its neighbouring pixels, which may not always hold true. To address this limitation, alternative loss functions are utilized to penalize structural differences between reconstructed regions and the corresponding regions in the original image [5].

The **gradient magnitude map** of an image is defined as:

$$g(\mathbf{I}) = \sqrt{(\mathbf{I} * \mathbf{h}_x)^2 + (\mathbf{I} * \mathbf{h}_y)^2}$$

where the image $\mathbf{I}$ is convolved with Prewitt filters of size 3:

$$\mathbf{h}_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \mathbf{h}_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Figure 9: Convolution with Prewitt filters is used for edge detection.

The **gradient magnitude similarity** map between the original image $\mathbf{I}$ and the reconstructed image $\mathbf{I}_r$ is then defined as:

$$GMS(\mathbf{I}, \mathbf{I}_r) = \frac{2g(\mathbf{I})g(\mathbf{I}_r) + c}{g(\mathbf{I})^2 + g(\mathbf{I}_r)^2 + c},$$

where c is a constant for stability in noisy areas [5].

The GMS is then enhanced by introducing a multi-scale variant (MSGMS), which operates across multiple image scales. An image pyramid is constructed, consisting of four different scales. This involves generating smoothed downsampled images at various scales through iterative average pooling, with a filter of size 2 and a stride of 2. The resulting image pyramid comprises images that are 1/2, 1/4, and 1/8 of the original size, in addition to the original image. The **MSGMS loss** is then computed as the mean value of the GMS distance map across these multiple scales:

$$L_G(\mathbf{I}, \mathbf{I}_r) = \frac{1}{4}\sum_{l=1}^{4} \frac{1}{N_l}\sum_{i=1}^{H_l}\sum_{j=1}^{W_l} 1 - GMS(\mathbf{I}_l, \mathbf{I}_{rl})_{(i,j)}$$

where $GMS(\mathbf{I}_l, \mathbf{I}_{rl})_{(i,j)}$ is a single pixel value of the GMS map between $\mathbf{I}$ and $\mathbf{I}_r$ at scale l. H_l, W_l and N_l are the height, width and number of pixels at scale l [5].

This multi-scale approach allows for a more comprehensive evaluation of structural similarity between images at different resolutions, enhancing the robustness and effectiveness of the loss function in capturing image similarities.

We define the **SSIM loss** as:

$$L_S(\mathbf{I}, \mathbf{I}_r) = \frac{1}{N_p}\sum_{i=1}^{H}\sum_{j=1}^{W} \mathbf{1} - SSIM\big(\mathbf{I}, \mathbf{I}_r\big)_{(i,j)}$$

where $SSIM\big(\mathbf{I}, \mathbf{I}_r\big)_{(i,j)}$ is the pixel-wise value of SSIM as in [12] and N_p is the total number of pixels.

The loss function we used for training the model, is a combination of MSGMS loss, SSIM loss and L2 loss (means squared error).

$$L = \alpha L_2 + \beta L_G + \gamma L_S$$

where α, β and γ are the respective loss weights.

After we are done with training we would like to execute the model on the test set, which contains images of banknotes both with and without anomalies, together with the ground truth masks, and we want to compute some anomaly score.

First, the anomaly map based on GMS is computed. Again following the multi-scale approach, beginning with the computation of the Gradient Magnitude Similarity GMS map between the input and reconstructed images across various scales. For each scale, a scaled GMS map is calculated by downsampling both the input image I_l and the corresponding reconstructed image I_{rl} to the scale l as it was done for the computation of the L_G loss [5].

Then for each scale l we upsample the $GMS(I_l, I_r l)$ maps to the original dimension. We then compute a multiscale GMS map $MSGMS(I, I_r)$ as the pixel-wise average of the rescaled GMS maps. We further convolve the $MSGMS$ map with a Gaussian filter f_G and subtract everything from a matrix of ones to obtain the anomaly map:

$$G(\mathbf{I}, \mathbf{I}_r) = \mathbf{1}_{H\times W} - (MSGMS(\mathbf{I}, \mathbf{I}_r) * \mathbf{f}_G)$$

The Gaussian smoothing is performed to avoid false positives from single-pixel peaks in the anomaly map. This also controls the tolerance of the size of the anomaly. Other types of smoothing filters could also be used.

Let us remember that we had a list of `k` values as we explained in Section 3.1. For each `k` in `k_list` we have a reconstruction and therefore we generate an anomaly map $G(\mathbf{I}, \mathbf{I}_r)_k$ for each one of them. We then take the average to obtain:

$$G_A(\mathbf{I}, \mathbf{I}_r) = \frac{1}{N_K} \sum_{k \in K} G(\mathbf{I}, \mathbf{I}_r)_k$$

We then take the maximum value as the anomaly score:

$$\epsilon(\mathbf{I}, \mathbf{I}_r) = \max(G_A(\mathbf{I}, \mathbf{I}_r))$$

and based on some threshold classify the image as anomalous or not. It's not clear from the paper [5] how to compute this threshold. In our implementation, we would use the `sklearn.metrics.precision_recall_curve` to compute precision, recall and thresholds, then choose the threshold that maximizes the F1 score.

4 Experiments & Results

4.1 Data set

50 Euro banknotes were used in our experiments, scanned with a resolution of 600 dpi. The images were then resized to 1024×1024 for convenience. Two datasets were created, one for each side of the banknote, and we also trained a model for each one of them. Both front and back datasets have 200 unique samples for the training and 35 unique samples for the testing.

The anomalies were generated on top of the 35 good test samples. For the anomaly dataset, due to the lack of a realistic dataset, Image Augmentation was used to generate the anomalies. The Python library Augraphy [16] was modified and used to generate anomalies that resemble pen-like scribbles and the ground truth binary masks. The modified version of the code can be found on GitHub [17]. The anomaly test set has 70 samples for each side. Some samples and their corresponding ground truths can be observed in Figures 10 and 11.

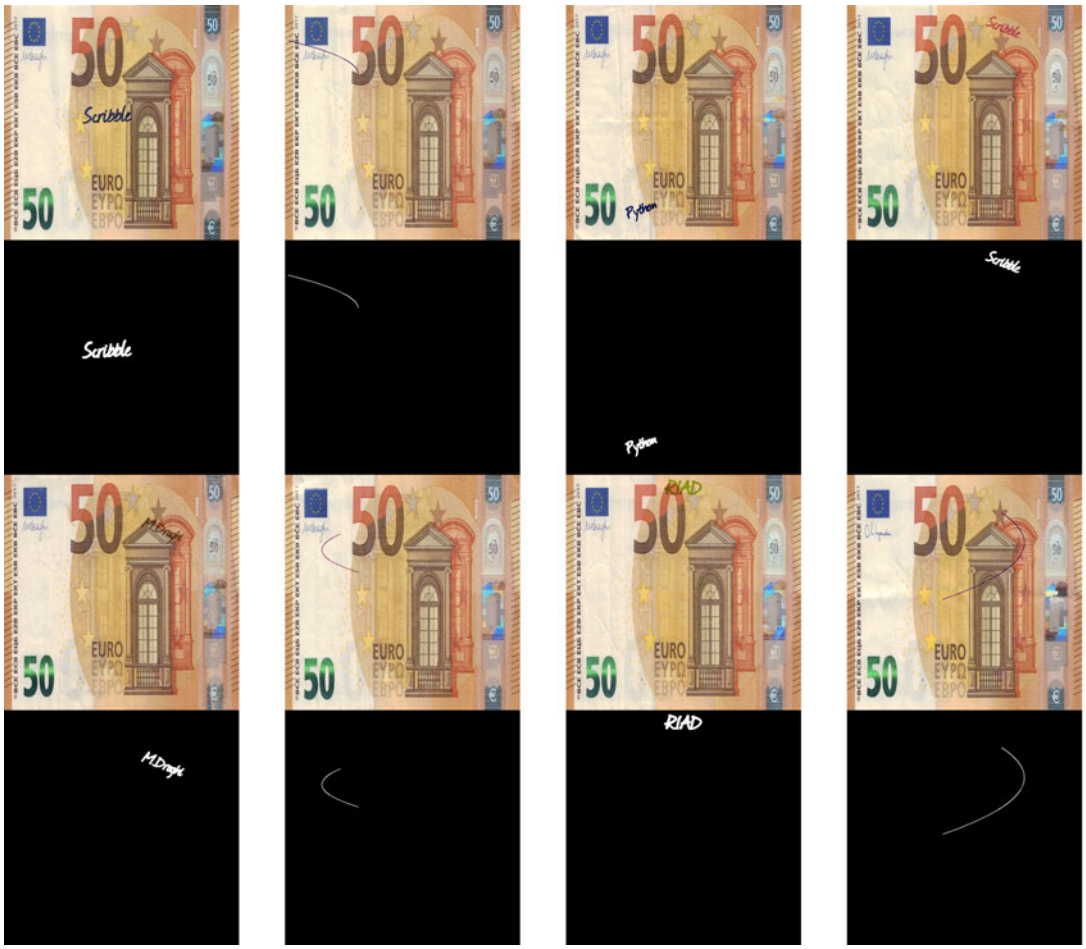


Figure 10: €50 front side samples from the anomaly test data set and their respective ground truth masks.

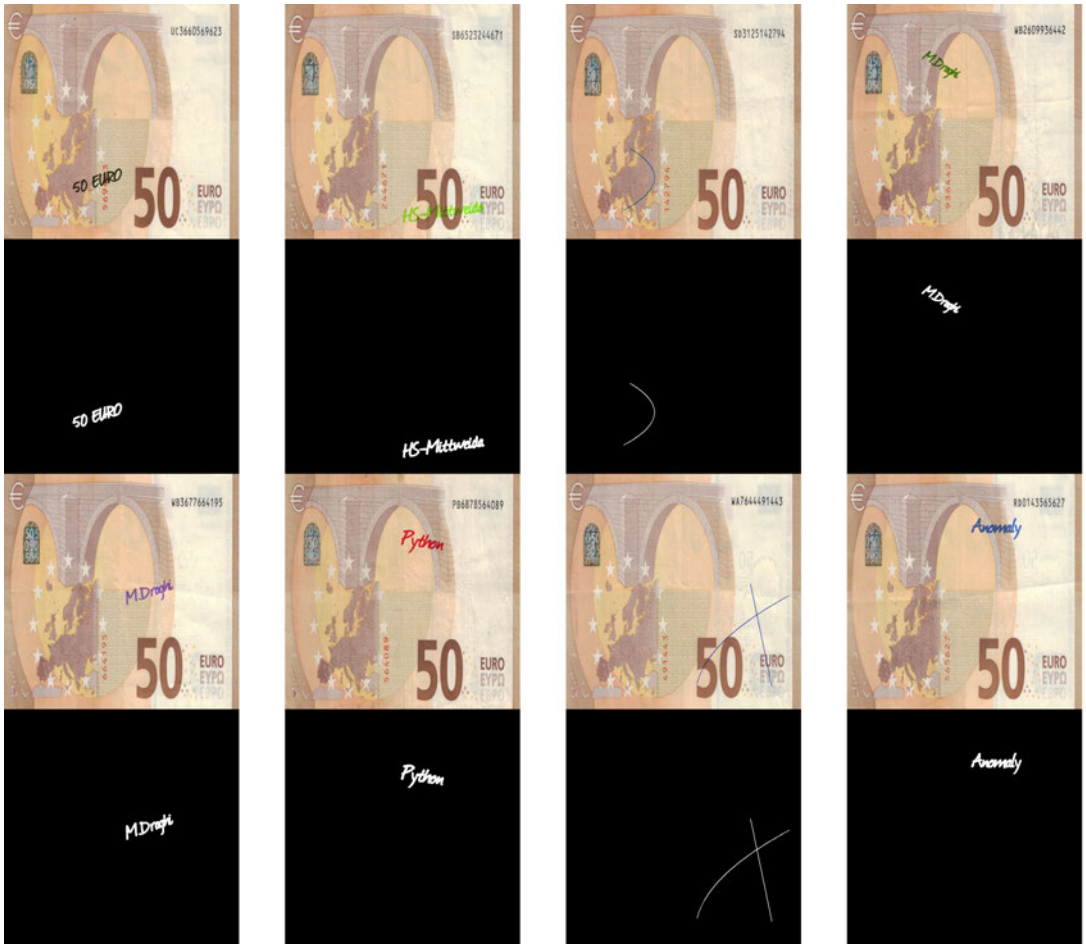


Figure 11: €50 back side samples from the anomaly test data set and their respective ground truth masks.

4.2 Training

The model was implemented with the PyTorch library. The training was done on a PC equipped with CPU AMD Ryzen 5600X, an internal SSD Samsung 970 EVO Plus 1TB M.2 and an NVIDIA RTX 3060 GPU with 12GB of VRAM. With CUDA support, we accelerated the training process by offloading computations to the GPU. The batch size was set to 8 to balance memory usage and computational efficiency.

The Adam optimizer was chosen for its effectiveness in training deep neural networks, offering adaptive learning rates and momentum. To prevent overfitting and ensure optimal generalization performance, an early stopping was implemented, halting the training if the validation loss did not improve for 20 consecutive epochs. The dataset was split into training and validation sets using an 80/20 ratio.

Typically, training sessions lasted around 5 hours, with early stopping commonly interrupting the training between 100-120 epochs to prevent overfitting and ensure efficient resource utilization. An exception can be seen in Figure 12 where the training didn't stop for 170 epochs. Nevertheless, it can be observed that the descent is slow after epoch 100.

For the loss function, a combination of Mean Squared Error (squared L2 norm) loss, Structural Similarity Index (SSIM), and Gradient Magnitude Similarity Deviation (GMS) was

used as explored in the section 3.3, where we also mentioned some loss weights α, β and γ. We left those weights equal to 1, so they had no impact. But perhaps SSIM loss could be balanced with those weights, since it has the most impact on the whole loss function, as we can see in Figure 12. We didn't focus much on experimenting with these parameters and in general on the training part since the reconstruction algorithm was performing very well as we will show in the next subsection.

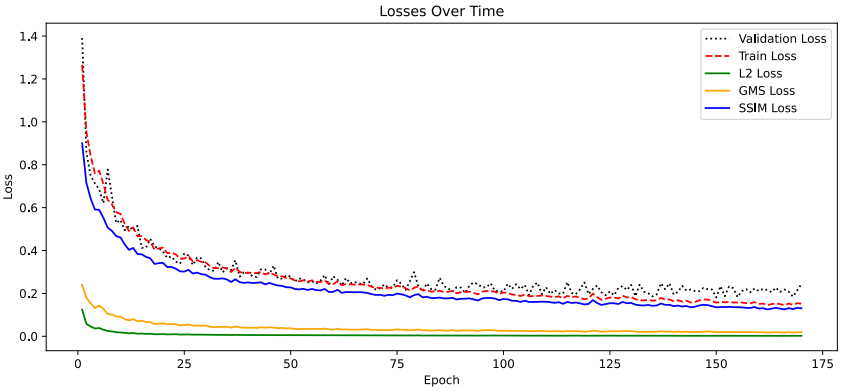


Figure 12: A training that went on for 170 epochs.

4.3 Results

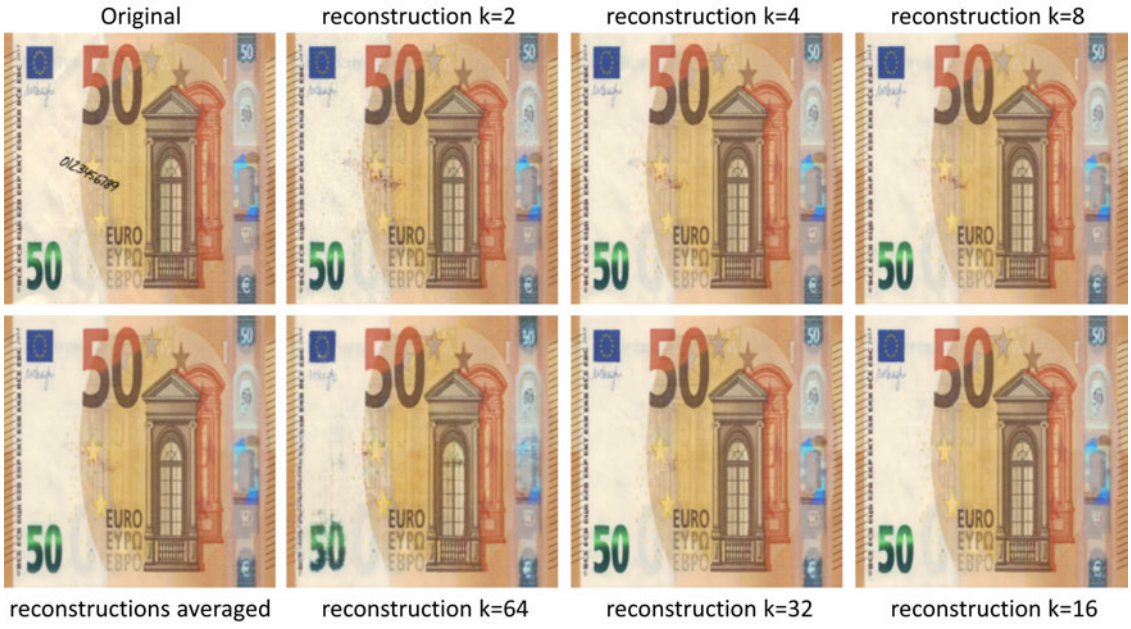


Figure 13: A 50 Euro front side anomalous sample and the corresponding reconstructions using the U-Net for different sizes k of masked regions.

After the model was trained, it was of interest to see the quality of the reconstruction. An initial visual inspection might bring us to the conclusion that masking with `k = 16` yields the cleanest reconstruction. We can not include the whole dataset in this document, but we confirm that by scrolling through the whole reconstructed dataset, it's noticeable that

the model is dealing very well with light-reflecting features and the relative positioning of the various components. However, it's important to maintain the focus on our primary objective of accurately identifying anomalies within the dataset. Upon generating the MSGMS maps for the same sample image, it becomes evident that augmenting the `k` values results in an escalation of noise, consequently amplifying the incidence of false positives. Therefore, using a `k_list = [2, 4, 8, 16]` may be a good approach, or alternatively, using a more condensed list or even a singular `k` value. The use of a different `k_list` and its impact on the performance is explored more deeply in Section 4.5.

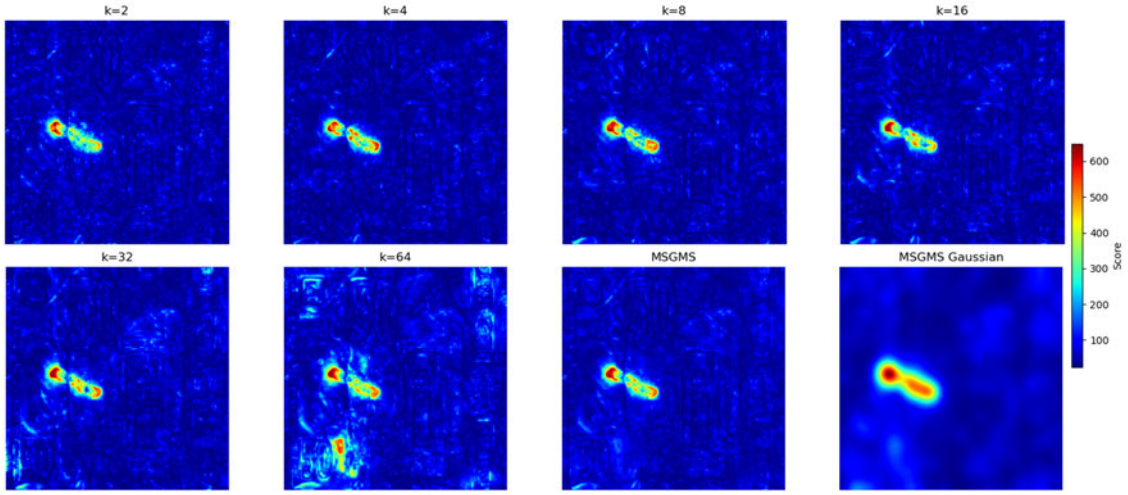


Figure 14: MSGMS score maps for different masked regions sizes k, MSGMS map combined and MSGMS map combined and smoothed with a Gaussian filter, for the sample in Figure 13.

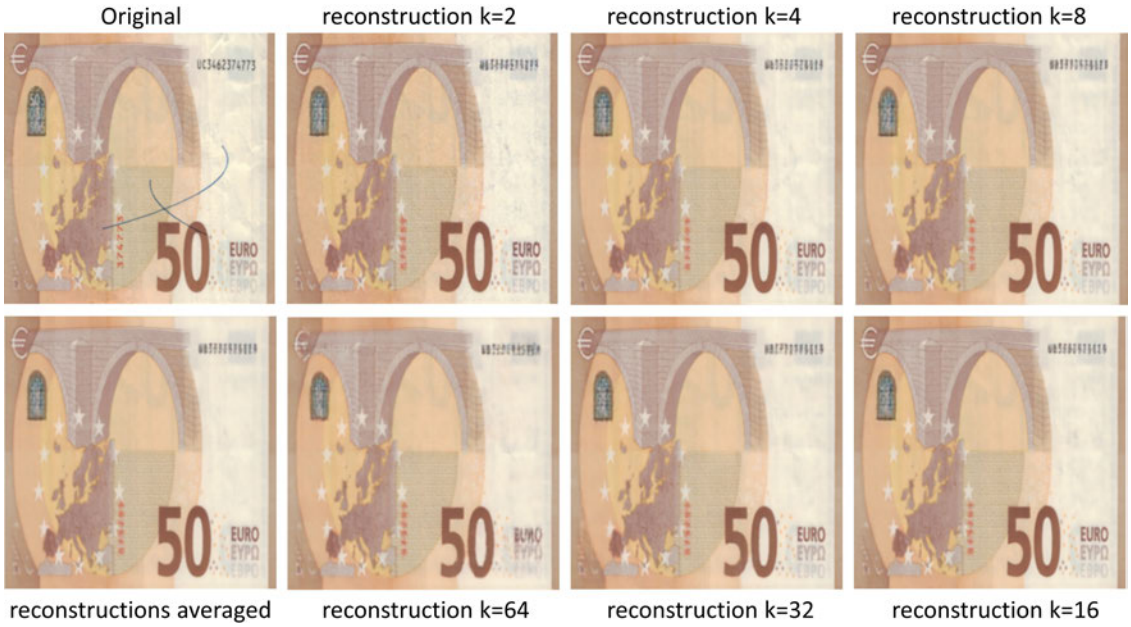


Figure 15: A 50 Euro back side anomalous sample and the corresponding reconstructions using the U-Net for different sizes k of masked regions.

For the back side, the reconstructions are also looking good at first as shown in Figure 15.

The two serial numbers and the transparent window appear to be problematic but upon the inspection of the MSGMS score maps in Figure 16, they don't seem to add too much noise to the score map. As observed for the front side, increasing the masked region size k results in noisier individual score maps.

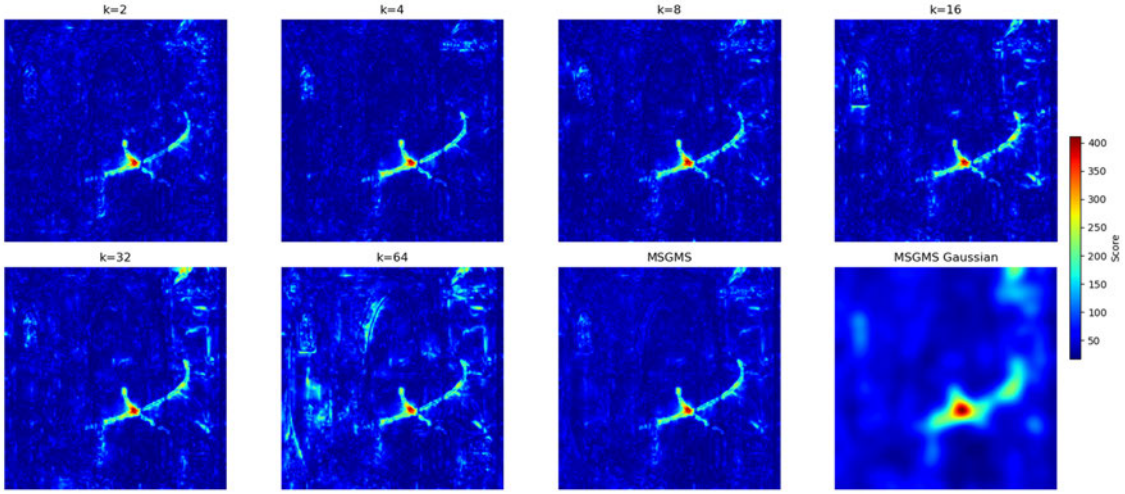
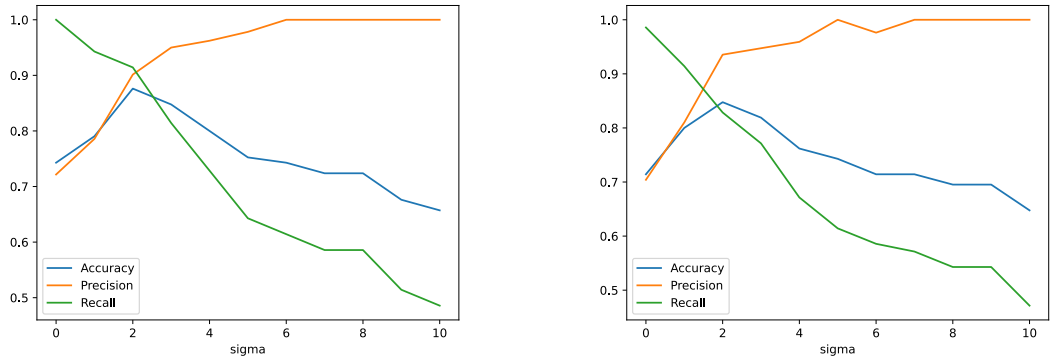


Figure 16: MSGMS maps for different masked regions sizes k, MSGMS map combined and MSGMS map combined and smoothed with a Gaussian filter, for the sample in Figure 15.

Gaussian smoothing is applied to deal with the noisy peaks in the anomaly map. It is a convolutional operation with a Gaussian kernel. The `scipy.ndimage.gaussian_filter()` function takes a sigma parameter which stands for stands for the standard deviation. We ran some testing on the whole test dataset, for sigma in the range of 0 to 10, to decide the ideal value for the Gaussian smoothing. We used the `k_list = [2, 4, 8, 16]` for this test.



(a) Front side

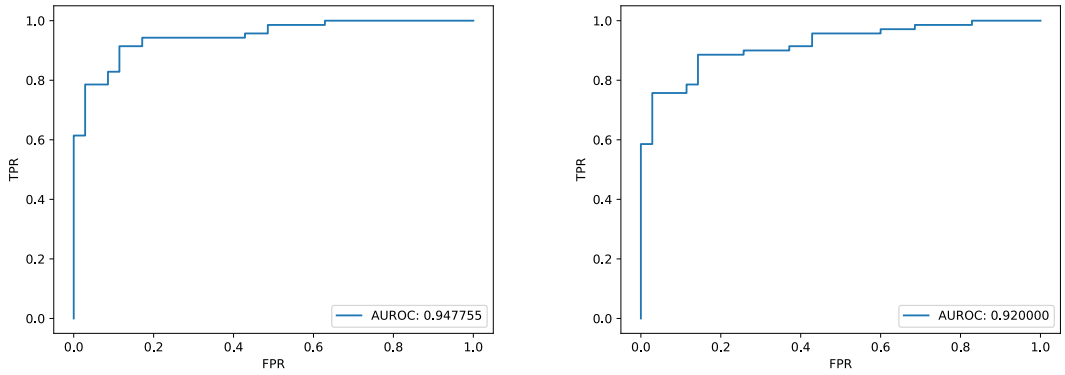
(b) Back side

Figure 17: Impact on the accuracy of Gaussian smoothing on MSGMS score map for different sigma values.

The results in Figure 17 suggest that setting the `sigma = 2`, results in the best accuracy for both sides. A trade-off between precision and recall can also be observed. It's crucial to mention that these results apply to our test dataset. As presented in Section 4.1, as anomalies we had the scribbles and the thin lines. For `sigma > 2` those thin lines tend to

lose their value on the anomaly map. But for spatially more present anomalies, a higher sigma would result in better accuracy.

After setting the `sigma = 2` and `k_list = [2, 4, 8, 16]`, running the inference for the whole test set yields the following results:



(a) Front side

(b) Back side

Figure 18: ROC curve.

Confusion Matrix				
	Front		Back	
	Predicted		Predicted	
	N	P	N	P
Actual N	30	6	30	5
Actual P	5	65	9	61

As a classification task, the metrics suggests a good performance. Upon a careful visual inspection of every single score map, it's clear the necessity for a better similarity assessment method. For the thin lines type of anomalies, the score is often above the threshold just in a very small portion of the whole line, a similar issue happens for bigger scribble-type anomalies if the background color is similar to the anomaly ink color. Furthermore, banknote folds often give rise to false positives and our dataset doesn't even contain so many banknotes with prominent folds. Considering that in a more realistic scenario, banknotes in circulation would have heavier folds, this similarity measure would be problematic for such an application case.

4.4 Time Performance

We don't know what kind of computational units are modern ATMs equipped with and how would this algorithm be implemented in a real-case scenario, but in any case, it would be of interest to see which parts of our implementation take the most amount of time. Time elapsed was measured on important snippets of the code in 3 different scenarios. In the following table, it can be observed that `torch.utils.data.DataLoader` costs a lot of time.

Inference on GPU						
	Single image		Test Set `batch_size = 1`		Test Set `batch_size = 30`	
Operation	time [s]	%	time [s]	%	time [s]	%
model load	0.33	6	0.33	1	0.33	1
data load	4.4	82	16	26	17	44
masking	0.19	3	19.6	32	1.4	4
U-Net	0.2	4	3.5	6	0.93	2
demasking	0.18	3	20	33	18.8	48
MSGMS score	0.07	1	1.76	3	0.58	1
Total	5.38	100	61.4	100	39.2	100

Running the inference on the whole test set shows that there was a difference in performance between using a batch size of 1 versus a batch size of 30. Specifically, the masking process improves significantly with higher batch size.

```
inputs = [data *
        (torch.tensor(mask, requires_grad=False).to(device)) for mask in Ms]
```

While the demasking line of code does not benefit from increased batch size:

```
output = sum(map(lambda x, y: x *
        (torch.tensor(1 - y, requires_grad=False).to(device)), outputs, Ms))
```

In case the ATMs are not equipped with a GPU, but some processing unit that's closer to a CPU, the following results could be of interest:

Inference on CPU				
	Single image		Test Set `batch_size = 30`	
Operation	time [s]	%	time [s]	%
model load	0.36	2	0.36	0
data load	4.44	28	17	1
masking	0.18	1	0.85	0
U-Net	11	68	1257	97
demasking	0.003	0	0.2	0
MSGMS score	0.15	1	15.6	1
Total	16.1	100	1291.4	100

The reconstructions for the whole data set through the U-Net took around 21 minutes. But surprisingly the whole demasking process took a fraction of a second. Perhaps the `sum(map(lambda ...))` function works better on the CPU. In both cases, the data loader is a bottleneck.

4.5 Masked Region Size

Motivated by the increasing noise in the anomaly maps for larger `k` values as presented in Figures 14 and 16, and by the high time complexity, experiments were conducted for different `k_list` values. For each `k_list` value, 10 inference cycles were executed, the AUROC score was computed and then the average of the 10 cycles was taken. The results are plotted and presented in Figures 19 and 20.

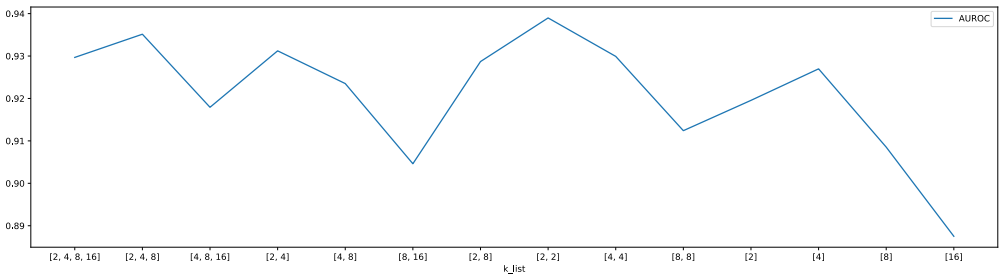


Figure 19: AUROC for different `k_list` values on front test set.

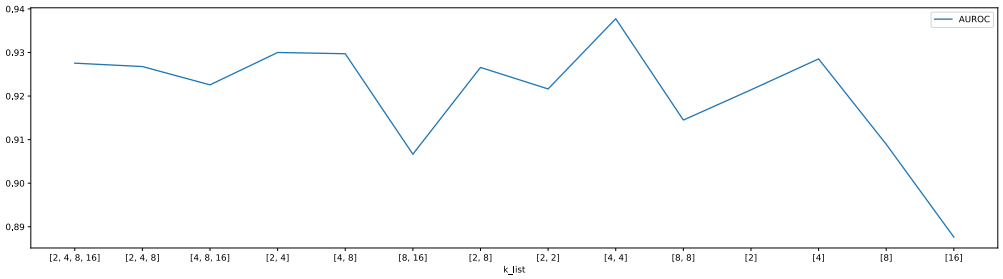


Figure 20: AUROC for different `k_list` values on back test set.

Different lengths and combinations of values from the set $\{2, 4, 8, 16\}$ were used. The y-axis indicates a small range $(0.89 - 0.94)$ of fluctuations of the AUROC score for different `k_list` values, nevertheless, it's clear from this experiment that the masked region size `k = 16` delivers the poorest performance from the set $\{2, 4, 8, 16\}$. Using the same `k` value twice indicates to deliver the best performance. And most interestingly a single value `k_list = [4]` is very close to the top-performing values.

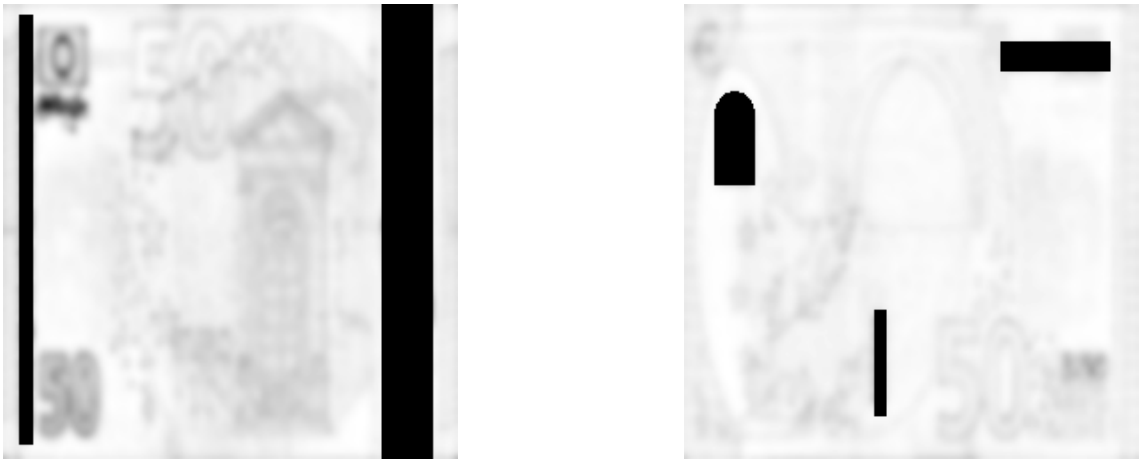
Time performance			
	`k_list = [2,4,8,16]` time [s]	`k_list = [4]` time [s]	Improvement [%]
(GPU) single image	5.38	5.04	7
(GPU) test set	39.2	23.07	70
(CPU) single image	16.1	7.7	109
(CPU) test set	1291.4	337	283

4.6 HSV Difference

During all the experiments, the slightly complicated MSGMS and its not satisfying performance led us to explore simpler methods. After converting both the original I and the reconstructed image I_r to the HSV color space, a pixel-wise difference revealed that complex is not always the best. Over the different trials, the following measure appeared to give promising results:

$$d_{HSV} = \sum_{c=1}^{3} |I^c - I_r^c| * \mathbf{f_G}$$

where the summation is over the 3 HSV color channels and $*\mathbf{f_G}$ is the Gaussian filter with `sigma = 2`. However, there was a problem with the light-reflecting features and the serial numbers: they were making a lot of noise in the anomaly score. The decision was then to reduce the influence in the anomaly map of those components by applying a weight map W. This map is calculated by running the inference on the whole train set i.e. clean banknote images, and computing the average d_{HSV}. This is a way to capture the banknote regions with the highest reconstruction variance. The idea is to multiply each channel with this weight map before the summation in the d_{HSV}, but this alone didn't solve the high rate of false positives. The weight map is then further edited manually, and all the pixels corresponding locally to the hologram stripe, the transparent window, the serial numbers, and the ECB (European Central Bank) written in different languages along the front left border, are set to zero. The weight map is a matrix corresponding to the image size, with continuous values in the range $[0, 1]$. Their visualization is in Figure 21.



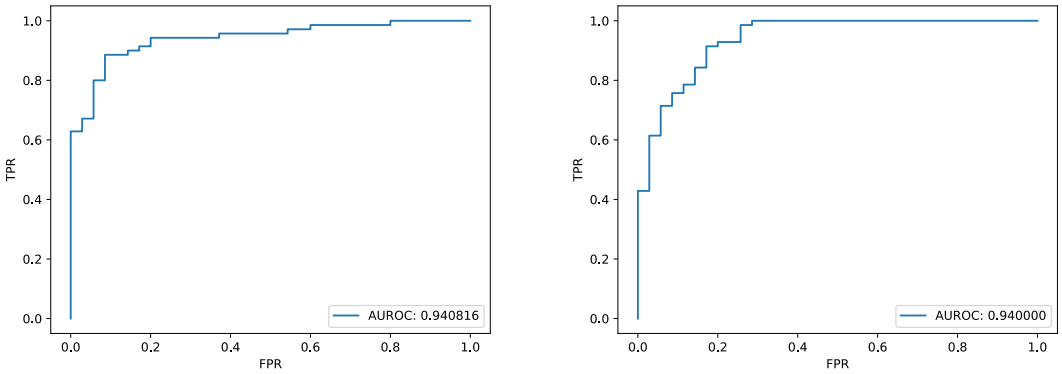
(a) Front side (b) Back side

Figure 21: Weight maps. Darker pixels correspond to values closer to zero.

Finally, the weighted similarity measure based on HSV color space is defined as:

$$d_{HSV}^W = \sum_{c=1}^{3} (|I^c - I_r^c| * \mathbf{f_G}) \cdot W$$

After setting the new way of computing the anomaly score and running the model on the test data set for a single masked region size `k = [4]`, the following results are presented:



(a) Front side (b) Back side

Figure 22: ROC curve - HSV Difference.

Confusion Matrix				
	Front		Back	
	Predicted		Predicted	
	N	P	N	P
Actual N	28	7	25	10
Actual P	5	65	1	69

If we look at the AUROC scores in Figures 18, 22 and confusion matrices of both, MSGMS and HSV-based, ways of computing the anomaly score, they seem to not differ much. However, analysing the anomaly scores and the segmentation contours, case by case visually, it is clear that the HSV-based method has a superior performance. Some samples are displayed in Figure 24. The segmentation contours in red ink mark the boundary between the detected anomaly and the rest of the banknote.

As a reminder, in every inference F1 maximizing score is selected. Using the HSV difference reduces a lot of noise allowing for a lower threshold on the same dataset. In Figure 24 the threshold is 93 for MSGMS and 29 for the HSV difference, expressed in pixel value. Every pixel in the anomaly score above the threshold will be detected as an anomaly. An argument could be that if the weight map is applied in a similar fashion to the MSGMS anomaly score, that noise could be eliminated as well, but this is not the case from our observations. While the noise in HSV difference-based anomaly maps has predictable locations, the noise in the MSGMS anomaly maps is quite random and often due to folds.

Figure 23: Somme false positives when using the HSV difference as a similarity measure.

HSV difference-based method on the back side gave a lot of false positives mainly around the dark "EURO" script on the bottom right as in Figure 23. As we can see in Figure 21

(a) HSV Difference

(b) MSGMS

Figure 24: Anomaly score maps and the corresponding segmentation contours for some samples.

the EURO script is a darker zone, i.e. higher variance in the reconstruction. Some folded or missing corners gave some false positives as well. The false positives for the front side are mainly around the EU flag and the signature under it, as in Figure 21 those are some of the darkest areas, that we didn't set fully to zero.

4.7 Brightness Experiment

In several cases a false negative classification was due to the anomaly color being very close to the banknote color (background color). To investigate further into the problem, a separate data set was created from a single banknote. RGB color was picked at a specific location on the banknote, the color was converted to HSV color space and the same anomaly was applied for the whole value (brightness) range $[0 - 255]$. A small anomaly was used, as the beige star on the middle-left in Figure 25.

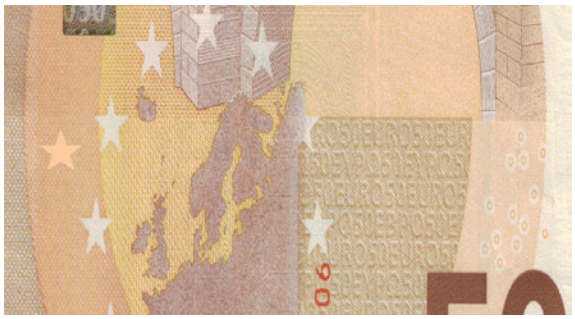


Figure 25: Anomaly star with Value channel = 255.

The threshold maximizing F1 score on the whole test set was then taken and fixed. As the similarity measure, the HSV Score was used. The Figure 26 shows the results.

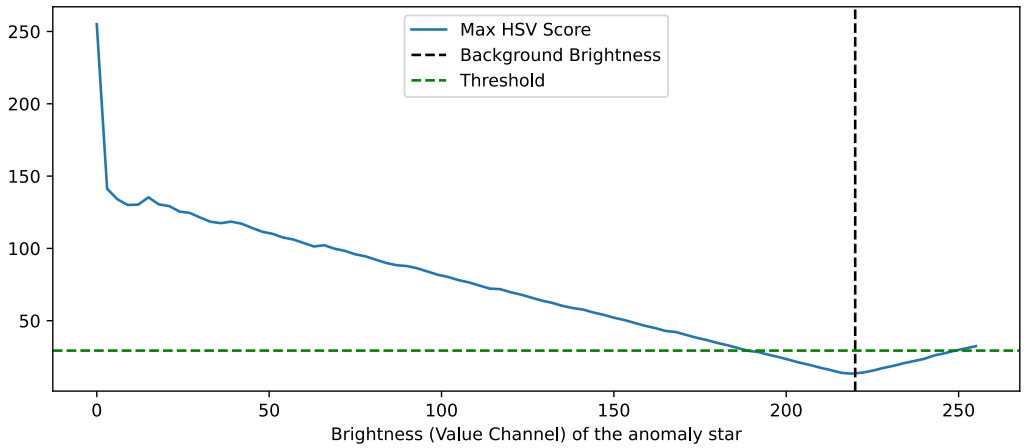


Figure 26: Brightness experiment - results.

5 Conclusion

The RIAD method shows interesting reconstruction performance and the MSGMS anomaly measure might perform very well for certain categories from the industrial set MVTec-AD. But banknotes tend to contain a lot of folds in random places, and once a banknote is reconstructed by inpainting, those folds are not present anymore. The MSGMS measure is based on the Prewitt filters, hence those folds are detected as edges therefore as structural components of the banknote. The fact that those edges are not present in both the original and the reconstructed images, yields a lot of noise in the anomaly score map.

HSV-Difference-based anomaly measure yields some undesired noise as well, but those locations are mainly the hologram stripe, the transparent window, and the serial numbers. Applying the weight map to the score eliminates this noise with the downside that those areas become vulnerable. Since certain areas in the weight map are set to zero, no anomalies will be detected in those locations. Furthermore computing those weight maps in our case was manual work which might be undesirable.

Regarding time performance we found that reducing the `k_list` to a single `k` value reduces greatly the time complexity without significant degradation in performance. The data loader takes a good portion of the total time hence that should be analysed and explored more in-depth. It should be clear the architecture of the processing unit to be used in whatever application scenario and all the codes should be adapted to that hardware.

A more realistic dataset should be created with well-defined unacceptable anomalies. In this study, no research was done on this aspect. Perhaps this study was focusing too much on small anomalies like the thin lines. If the task was just detecting big stains, the algorithm presented would perform very well by just increasing the effect of the Gaussian smoothing.

At the beginning of the study I only had a clear idea about the problem that had to be solved and the vague direction was to use some encoder-decoder-based approach. The RIAD was discovered casually and it seemed a promising method. After a test the reconstruction results seemed promising and the decision was to continue with RIAD. After gaining some academic maturity and domain knowledge, if I had to do it all over again, I would take a different path. I would advise exploring other more recent papers preferably selecting the top-performing papers on the MVTec-AD data set. A Benchmark list is presented on the Papers with Code website [18], where the RIAD paper is only in the 75th position.

I conclude this thesis with a question that arose during the research. How do you define mathematically what's socially acceptable? Perhaps a question that might be the biggest challenge even for the top-performing anomaly detection methods specifically applied to our banknote case. An anomaly resulting from wear and use discolouration might have the same mathematical weight as a message or symbol that's morally or socially unacceptable. This point makes anomaly detection on banknotes even a greater challenge.

References

[1] John E. Sandrock. "The Use of Bank Notes as an Instrument of Propaganda". In: (). URL: https://www.thecurrencycollector.com/pdfs/The_Use_of_Bank_Notes_as_an_Instrument_of_Propaganda_-_Part_I.pdf.

[2] Erwin Beyer. *China: Banknote mit Überdruck der Falun Gong*. URL: https://www.geldscheine-online.com/post/china-banknote-mit-%C3%BCberdruck-der-falun-gong.

[3] Mayke Blok. *I Have a Chinese Banknote That Everyone in China Is Scared Of*. URL: https://www.vice.com/en/article/3b7bjb/i-have-a-chinese-banknote-everyone-in-china-is-scared-od.

[4] European Central Bank. *How euro banknotes are produced*. URL: https://youtu.be/SKq1UnuVijA?t=211.

[5] Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. "Reconstruction by inpainting for visual anomaly detection". In: *Pattern Recognition* 112 (2021), p. 107706. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2020.107706. URL: https://www.sciencedirect.com/science/article/pii/S0031320320305094.

[6] Paul Bergmann et al. "The MVTec Anomaly Detection Dataset: A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection". In: *International Journal of Computer Vision* 129.4 (2021), pp. 1038–1059. DOI: 10.1007/s11263-020-01400-4.

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: (2015). arXiv: 1505.04597 [cs.CV].

[8] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Available online at http://szeliski.org/Book/. Springer, 2011, pp. 72–78.

[9] SharkDderivative. *HSV color model mapped to a cylinder*. URL: https://commons.wikimedia.org/wiki/File:HSV_color_solid_cylinder_saturation_gray.png.

[10] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep Learning*. MIT Press, 2016, p. 323.

[11] Vincent Dumoulin and Francesco Visin. *A guide to convolution arithmetic for deep learning*. 2018. arXiv: 1603.07285 [stat.ML].

[12] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[13] David M. W. Powers. *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*. 2020. arXiv: 2010.16061 [cs.LG].

[14] Francesco Visin Vincent Dumoulin. *Convolution arithmetic*. URL: https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md.

[15] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[16] The Augraphy Project. *Augraphy: an augmentation pipeline for rendering synthetic paper printing, faxing, scanning and copy machine processes*. Version 8.2.6. URL: https://github.com/sparkfish/augraphy.

[17] Kornelije Juric. *Data Augmentation - Banknote Images*. URL: https://github.com/korn3lie/data-augmentation-banknotes.

[18] *Papers with Code*. URL: https://paperswithcode.com/sota/anomaly-detection-on-mvtec-ad.

Eidesstattliche Erklärung

Hiermit versichere ich – Kornelije Juric – an Eides statt, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach Publikationen oder Vorträgen anderer Autoren entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt oder anderweitig veröffentlicht.

Erlangen, 21.04.2024
Ort, Datum

Kornelije Juric