Handwritten-notes Denoising

Olmo Baldoni, Cristian Bellucci, Danilo Caputo

Università di Modena e Reggio Emilia, Modena, Italia

325524@studenti.unimore.it, 322906@studenti.unimore.it,
246019@studenti.unimore.it

Abstract

In this project we provide tools to accelerate the collection of data needed to address mathematical equation detection and optical character recognition (OCR). Handwritten notes pose several difficulties for automatic extraction of text and equations, mainly because of the inherent noise created, for example, when notes are taken on lined or squared sheets, rather than on a blank sheet of paper.

Our goal is to have a CNN trained to remove square grids from handwritten notes and to obtain a text-only, noise-free image. For this purpose, we used UNet and a specially created synthetic dataset.

In addition, we present a simple approach to crop and warp images of hand-taken notes from photos, and a retrieval system to automate the task of collecting images that need to be processed by the denoising network. (GitHub Link: https://github.com/IloDan/Handwritten-notes-Denosing.git)

1 Introduction

The recognition of mathematical characters in handwritten notes poses several challenges. In addition to distortions introduced by the camera during image acquisition, the fundamental issue lies in the intrinsic noise present in the data.

Notes are often written on grid or lined paper, and for OCR of formulas, the papers themselves become a source of noise. Since characters written on the sheets overlap with the grid or line structures, their presence complicates the ability to use a model to detect and recognize them.

Therefore, in this project, we aimed to explore methods for automating the denoising operation of handwritten notes.

Initially, we attempted to use the fast Fourier transform to identify and filter frequencies corresponding to horizontal and vertical lines, but this approach requires ad hoc parameter settings for each type of note and also tends to remove text in conjunction with grids. For this reason, we trained a neural network to learn the various types of noise present in handwritten notes. In this way, denoising can be done on a wide range of notes fairly automatically, without the need to preprocess each individual image. We chose UNet as the network to implement the task of denoising grids and lines.

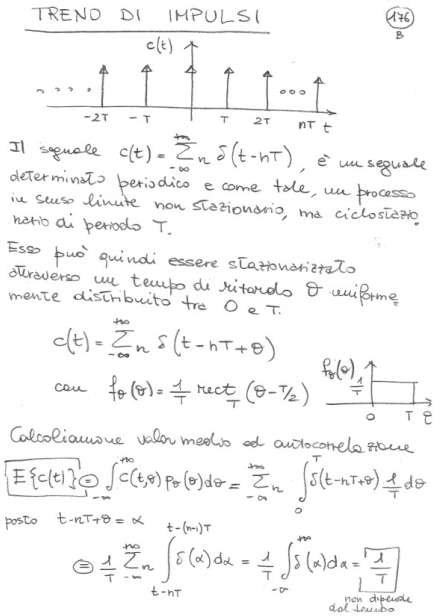
When photographs of notes show distortions due to perspective and/or unwanted elements in the background (portions of the surface on which the sheets are placed), data preprocessing is required. We used morphological operations along with Canny to extract the edges of the sheet from the image, allowing for subsequent correction through perspective transformations.

In addition, we have implemented an image retrieval system to identify notes that require background denoising, from those that are already suitable for the text recognition task. In other words, it allows us to automatically identify images that require processing. The system is also capable of grouping notes from the same classes of notes. In this way, we can retrieve an arbitrary number of similar images from a single image of notes.

2 Synthetic Dataset for denoising handwritten notes

The most important work has definitely been the generation of the Synthetic Dataset needed to train UNet to recognize and eliminate squares or rows from non-white sheets.

The Dataset consists of 132000 synthetic images obtained by a Data Augmentation process. For its creation, we collected 3300 images of handwritten notes on blank sheets (fig. 1a) and 70 templates of different types of squares or rows as background((fig. 1b). ¡



(a) Example of handwritten note

(b) Example of template

Figure 1: Images to overlay to get an example of the dataset

For each note, 40 of the 70 grids were randomly applied. The grids $G(x, y)$ were overlapped to the notes images $N(x, y)$ using the max operator over the negative gray scale images:

$$I_{neg}(x, y) = \max(G_{neg}(x, y), N_{neg}(x, y)) \tag{1}$$

$$I(x, y) = 255 - I_{neg}(x, y) \tag{2}$$

To make the images $I(x, y)$ more realistic, we introduced random sinusoidal distortions to the applied templates $G(x, y)$ so that the lines of the rows or squares were not precisely straight, but slightly curved, as when taking a picture of a sheet of paper.

The new coordinates of the pixels are computed as:

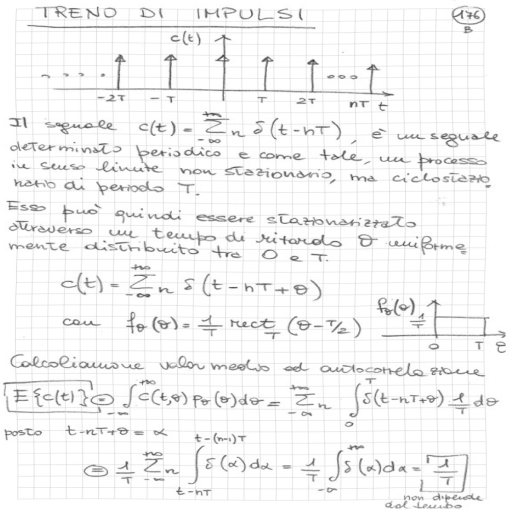
$$x_{new} = x + A \sin \frac{y}{k} \tag{3}$$

$$y_{new} = y + A \sin \frac{x}{k} \tag{4}$$

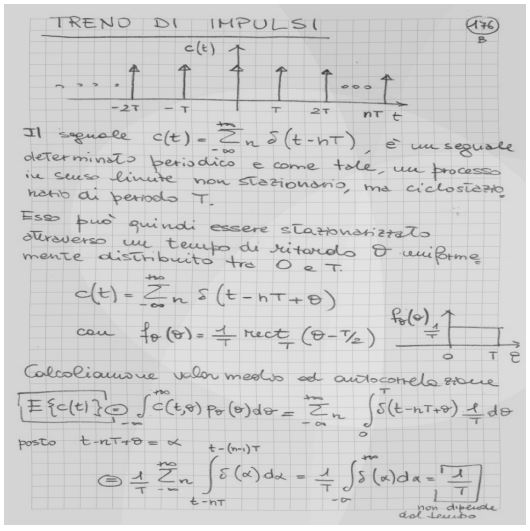
where $A \sim rand(1, 3)$, and $k \sim rand(50, 150)$ are the random parameters that regulate the distortion for each pixels.

Each pixels value $G'(x_{i_{\text{new}}}, y_{i_{\text{new}}})$ is calculated by interpolating bilinearly the neighborhood of the pixels in the not distorted image.

Finally, since in photos of real notes there are usually variations in brightness in several regions of the same image, we tried to simulate this by having elliptical masks randomly applied to one or more regions of the image to alter the brightness at the time of dataset generation for each sample. An example of an image generated before and after the brightness change is shown in Fig. 2, it is also provided the histogram of colors of the two images (Fig. 3)

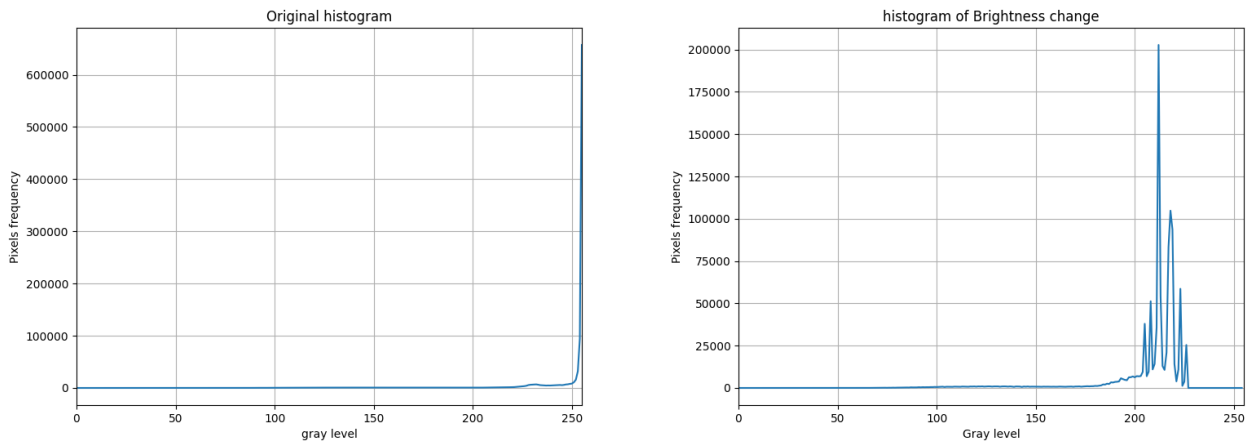


(a) Image without brightness changes



(b) Image with brightness changes

Figure 2: Dataset example before and after the brightness change



(a) HOC of Fig. 2a

(b) HOC of Fig. 2b

Figure 3: Histogram of color of images in Fig 2

3 Image cropping and warping

Usually the handwritten notes include scans and photographs taken by smartphones. Hence, in this section we present the way in which we crop a sheet of paper in a image, detecting its contours and discarding the surround, and warping it to original resolution.

The implementation involves detecting the edges of the document, identifying the outline representing the paper, and applying a perspective transformation to obtain a top-down view of the document.

3.1 Edge detection and Perspective transformation

To detect edges in the image, we use morphological operation combined with Canny algorithm. The algorithm itself involves several steps, including Gaussian filtering, gradient calculation, non-maximum suppression, and hysteresis thresholding.

Firstly, the image is converted from RGB to greyscale in order to simplify the computation required for edge detection.

Secondly, the image is blurred using a Gaussian filter to reduce noise and smooth out any irregularities in the image. The size of the kernel used affects the level of blurring applied to the image, and thus produce different results at the end of the edge detection pipeline. By adjusting the kernel size, the trade-off between noise reduction and edge preservation can be optimized, resulting in a cleaner and more accurate result.

Then the morphological operations are used to preprocess the images before using Canny [9] to detect the edge.

The morphological operation of opening [5], is used to remove noisy details and improve the connectivity of the pixels in the image, decreasing the false edge detection.

Instead, the Morphological Gradient, i.e., the difference between erosion and dilation on the image, provides an outline of the images, thus highlighting the contours of the sheet.

Next, Canny is applied and the result is a binary image with highlighted edges as shown in Figure 4.

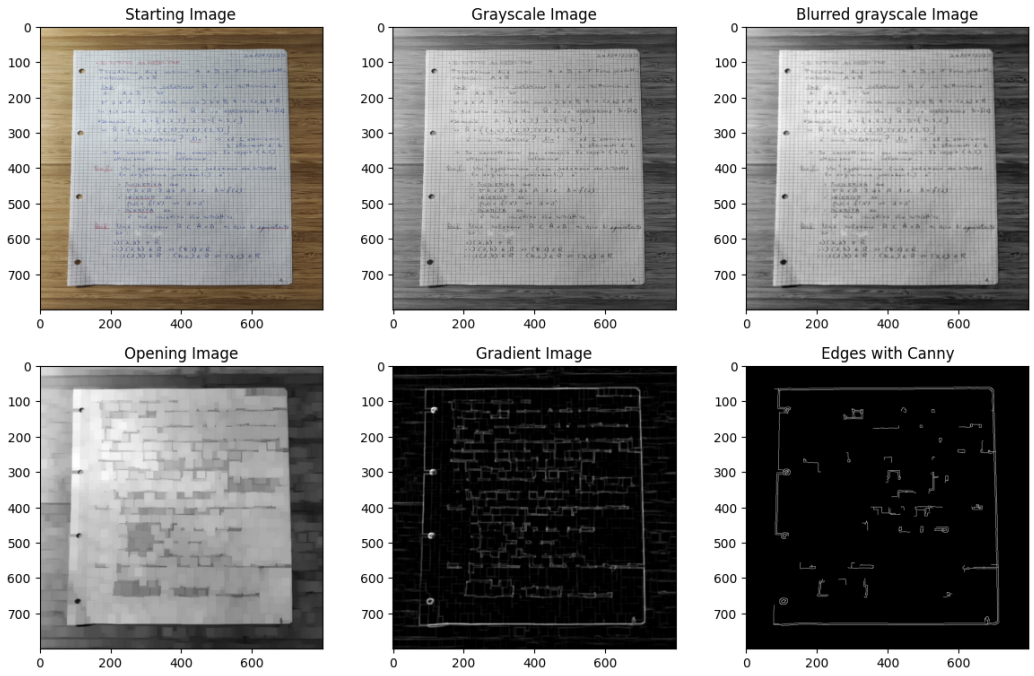


Figure 4: Pipeline of edge detection

After finding all the possible edges, the one with the largest area is considered the page contours, and its corners are used for perspective transformation. [6]

Specifically, once the corners of the page have been obtained, the transformation matrix defining the mapping between the corners found and those of the source image can be calculated.

Once this matrix is found, it is possible to warping the image using the Perspective Transform as illustrated in Fig 5.

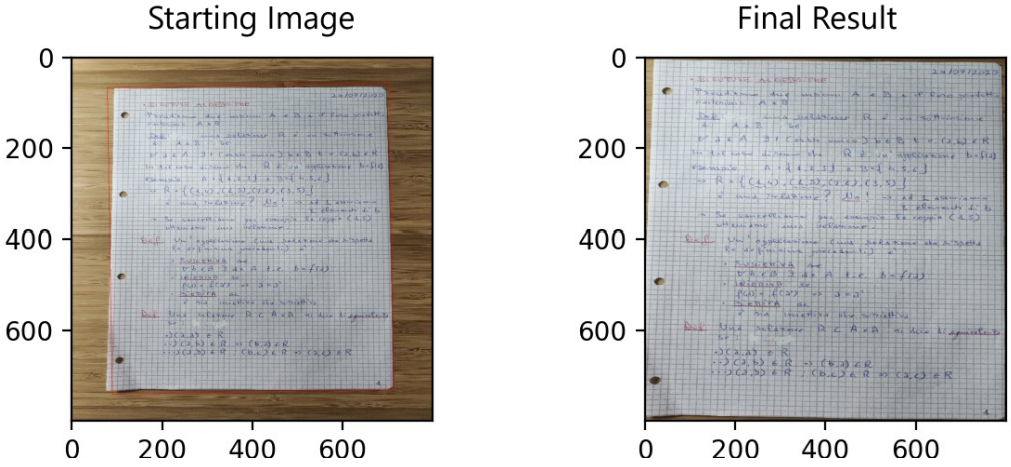


Figure 5: From starting to dewarped image

4 Architectures and Results

4.1 UNET

In our pursuit of finding an effective solution for grid removal, we explored various alternative methods, one of which was the Fourier Transform. While the Fourier Transform showed promise in efficiently removing grids from images, it posed a significant challenge. The primary issue we encountered was the need to adjust the parameters of the Fourier Transform method on a per-image basis. This manual parameter tuning made it impractical and rendered automation of the process virtually impossible.

The challenge of having to tailor parameters individually for each image was a significant bottleneck in our workflow. It not only consumed a substantial amount of time but also limited the scalability and generalizability of the solution. As a result, we decided to shift our approach toward deep learning.

The choice of using the UNet architecture to remove grids from clipboards depends on the specific needs of the problem and the nature of the images involved. UNet is a convolutional neural network (CNN) architecture commonly used for segmentation problems, where the goal is to identify and separate different regions or objects within an image. The main reason we chose it is the downsampling and upsampling protocol; the architecture was designed to effectively downsample and upsample information in images. This is critical to solving the grid removal problem, as it requires the ability to preserve important image details. UNet is known to preserve detailed details because of the skip connections between the encoder and the decoder. These connections allow the model to retrieve low-level information that might otherwise be lost, which is especially useful for maintaining clarity of text and fine details in clipboards. Another reason is the effectiveness for segmentation problems, UNet was originally developed for biomedical segmentation problems, where the goal is to identify and separate regions of interest in medical images. This demonstrates its effectiveness in separating regions in complex images, including the case of grid removal. The model is seen below of the U-net architecture.

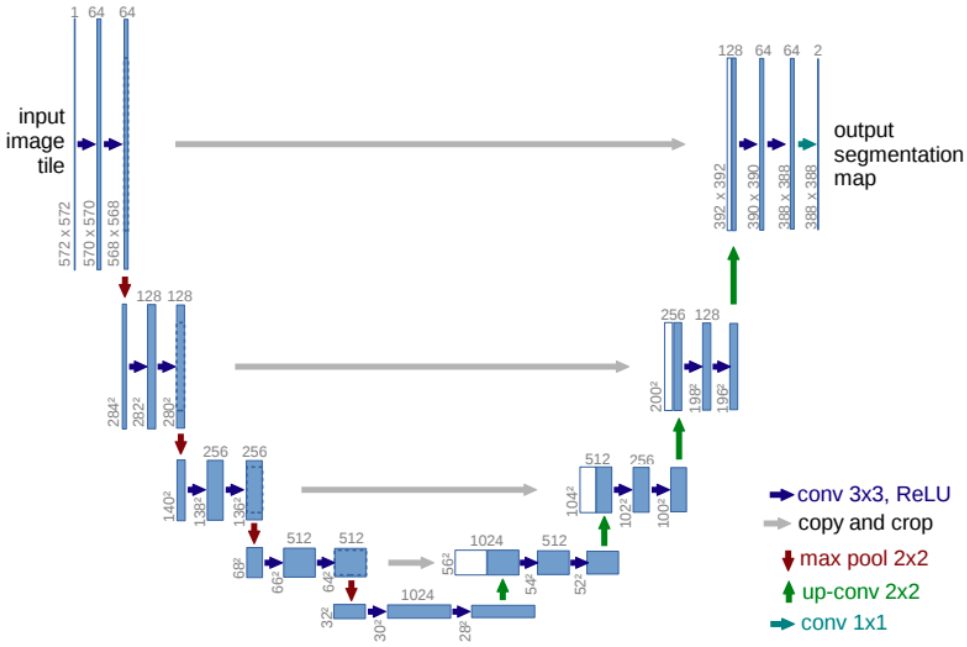


Figure 6: U-net architecture [1]

In our pursuit of training a robust model for grid removal, we conducted several training iterations using a dataset we created, which was divided into a 2/3 train and 1/3 test split. We experimented with three different resolutions for the images, specifically 256, 512, and 1024 pixels. Ultimately, we settled on using the 1024-pixel resolution because the lower resolutions lacked the necessary image definition.

Initially, our dataset consisted of 1700 images, each containing 30 synthetic grids. However, we encountered a significant challenge during training, as the model quickly began to exhibit signs of overfitting. It became apparent that the model was primarily learning to remove the synthetic image grids present in the dataset, which was not our desired outcome.

To address this issue, we decided to increase the number of grids in the dataset. This adjustment helped alleviate the overfitting problem to some extent. However, we observed that the network still struggled when it came to removing grids from real images, which remained a critical objective.

As a solution, we implemented the data augmentation techniques previuosly decribed. Additionally, we expanded the dataset by increasing both the number of base images and the variety of grids. This comprehensive approach ultimately resulted in the creation of our final dataset.

These iterative steps and adjustments in dataset composition were crucial in our journey to train a model that could effectively remove grids from real-world images while maintaining its generalization capability.

In order to effectively train the network with such large images (1024x1024), we adopted a distributed training approach. This allowed us to handle the large amount of data and computation required to train the network efficiently. In this process, we divided the workload between two computational nodes, allowing for a distribution of resources and parallelization of computations. In this way, we were able to make the best use of available resources to

successfully train the neural network by addressing image size challenges.

During the training process of the UNet model, we used the Mean Squared Error (MSE) loss function. To monitor the training progress and the model's ability to generalize, we averaged the loss on both the training set and the validation set at each epoch. We chose MSE as the loss function, which measures the average squared difference between the pixel values of the denoised image and its ground truth (GT) image [2] [3]. The MSE loss is a simple and effective way to compare the similarity between two images, and it is widely used in image processing and computer vision tasks [2] [4] The Mean Squared Error (MSE) is calculated as follows:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{I}(i) - I(i)\right)^2 \tag{5}$$

Where:

$$\begin{aligned} MSE &: \text{Mean Squared Error} \\ N &: \text{Total number of pixels in the images} \\ \hat{I(i)} &: \text{Pixel value in the flattened target image} \\ I(i) &: \text{Pixel value in the generated flattened image} \end{aligned}$$

Two optimizers, SGD and Adam, were utilized. SGD was employed in conjunction with a onecycle scheduler and a momentum of 0.9. This choice was informed by the literature, as SGD is well-known for its slowness and susceptibility to becoming trapped in local minima. However, by introducing momentum and a scheduler, we expedited the convergence process. In contrast, Adam was employed independently due to its inherent ability to automatically adjust its learning rate. Ultimately, we opted for SGD due to its lower loss and superior inference results. Due to memory limitations, we used a batch size of 2 per process, with a total of 2 processes, and the initial learning rate was set to 0.01, which was subsequently dynamically adjusted using a onecycleLR approach with a maximum learning rate (maxLR) of 0.3. This means that during the training process, the learning rate was modified dynamically, allowing it to vary within a predefined range, with the upper limit (maxLR) capped at 0.3. On average, we performed training for about 5 epochs. However, we implemented periodic saving of the model at each epoch and then selected the model with the best performance based on validation loss. This allowed us to select the model with the optimal performance rather than relying on a fixed number of epochs.

These configurations were chosen to optimize the training of the UNet model to ensure the convergence of the model to an effective solution for removing grids from images.

4.2 Experiments

In this chapter we will examine in detail the experiments conducted in the context of our project. Experimental methods, results and conclusions from this crucial phase of our work will be presented. To assess the performance of the three models, we employed Mean Squared Error (MSE) and the Structural Similarity Index (SSIM) metrics. [4] The MSE is obviouvsly the same of the formula (5). The SSIM (Structural Similarity) index is a method for comparing the similarity between two images. It is a full reference metric, in other words, the measuring of image quality based on an initial uncompressed or distortion-free image as a reference.

$$\text{SSIM}(\hat{I}, I) = \frac{(2\mu_{\hat{I}}\mu_I + C_1)(2\sigma_{\hat{I}I} + C_2)}{(\mu_{\hat{I}}^2 + \mu_I^2 + C_1)(\sigma_{\hat{I}}^2 + \sigma_I^2 + C_2)}$$

(6)

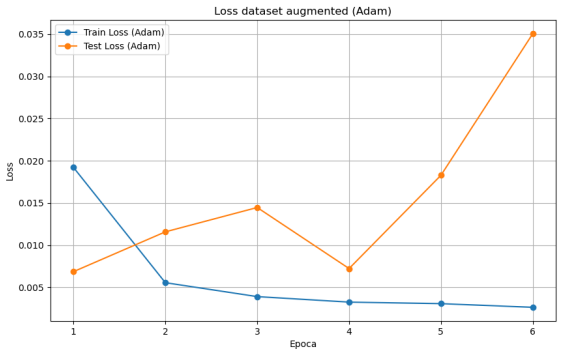
In the SSIM formula:

- $\hat{I}$ is the GT and I the image to be evaluated ;
- $\mu_{\hat{I}}$ and μ_I are the average of $\hat{I}$ and I respectively.
- $\sigma_{\hat{I}I}$ is the covariance of $\hat{I}$ and I.
- $\sigma^2_{\hat{I}}$ and σ^2_I are the variances of $\hat{I}$ and I respectively.
- C_1 and C_2 are two variables to stabilize the division with weak denominator.

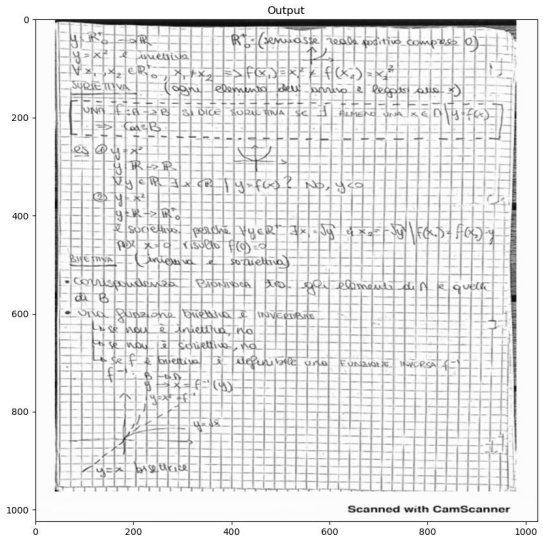
The tables 1, 2, 3 compare the calculated metrics to compare the difference between the denoised image and the ground truth versus the ground truth to which the grid is synthetically applied and GT. The figures in the tables are in the appendix. Although these evaluation methods are not perfect, the most effective way to assess model efficiency remains visual inspection.

4.2.1 First Trial

Initially, as mentioned above, we trained the model on a dataset without augmentation and the Adam optimizer. The first image shows the loss during training, and the second displays some example images.



(a) Loss during training (No Augmentation)



(b) Example images (No Augmentation)

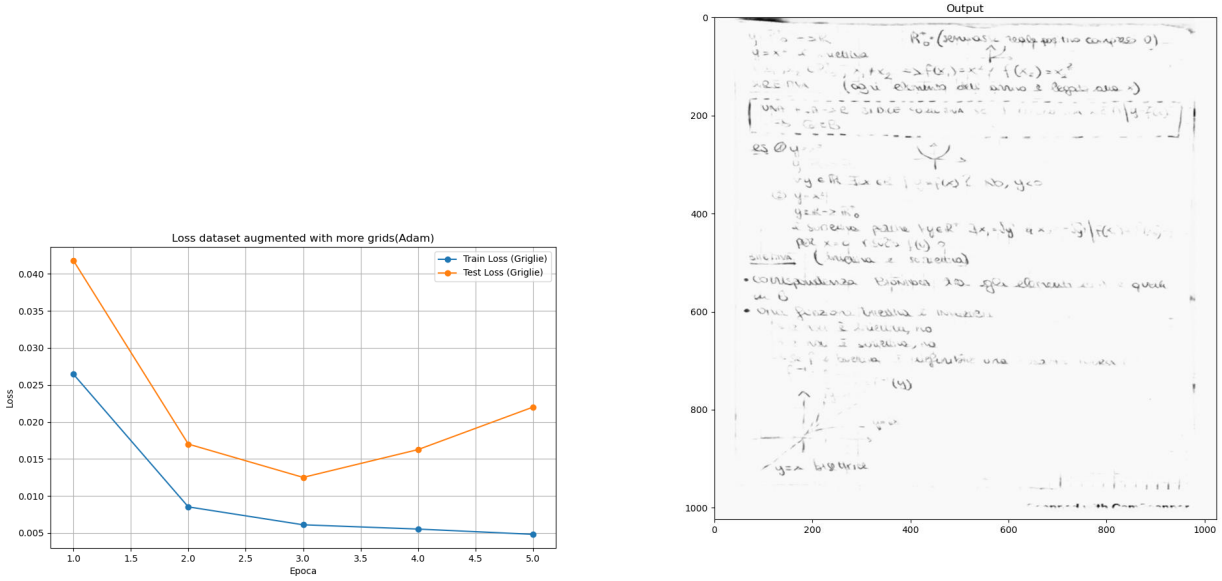
Figure 7: First Trial

Table 1: First trial

Figure	MSE		SSIM	
	Inferenced	Original	Inferenced	Original
16	0.0502	0.0063	0.4565	0.6743
17	0.0683	0.0981	0.3917	0.1724
18	0.0568	0.0924	0.4668	0.1598

4.2.2 Second Trial

In this phase of the experiment, we used the augmented dataset always with Adam, resulting in noticeable improvements in loss minimization.



(a) Loss during training (With Augmentation) (b) Example images (With Augmentation)

Figure 8: Second Trial

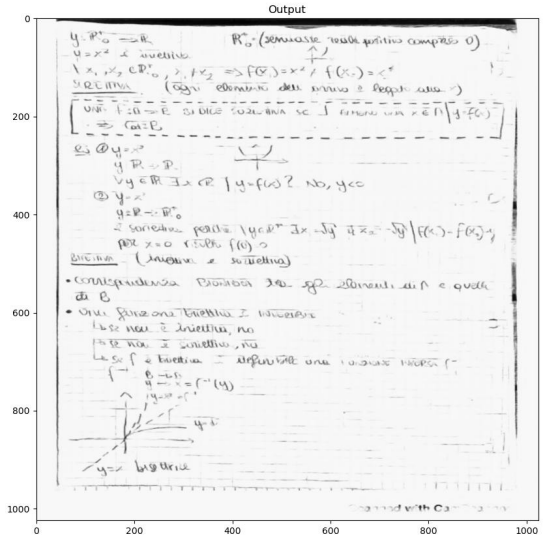
Table 2: Second trial

Figure	MSE		SSIM	
	Inferenced	Original	Inferenced	Original
19	0.0576	0.0063	0.4685	0.6743
20	0.0678	0.0981	0.4319	0.1724
21	0.0573	0.0924	0.4749	0.1598

4.2.3 Third Trial

For the third trial, we switched to the SGD optimizer, with a momentum of 0.9, and employed a oneCycleLR scheduler. The dataset used was still augmented. This change led to a significant performance improvement.

(a) Loss during training (SGD Optimizer)



(b) Example images (SGD Optimizer)

Figure 9: Third Trial

Table 3: Third trial

Figure	MSE		SSIM	
	Inferenced	Original	Inferenced	Original
22	0.0580	0.0063	0.4672	0.6743
23	0.0691	0.0981	0.4424	0.1724
24	0.0582	0.0924	0.4745	0.1598

5 CBIR for Handwritten Notes

Content-Based Image Retrieval (CBIR) systems retrieve images with similar content to a query image by comparing visual features such as color, texture, or shape. CBIR systems use a distance metric like Cosine Similarity to determine the similarity between the query and database images. Recently, Convolutional Neural Networks (CNNs) have been used for CBIR, enabling the learning of rich local and global features from raw image data. CBIR systems are crucial in various applications, including search engines, e-commerce, and medical imaging. In

handwritten notes denoising, a CBIR system can retrieve images of similar notes, reducing the denoising process's complexity. In this chapter, we present our image retrieval system utilizing Regional Maximum Activation of Convolution (RMAC) vector descriptors. Our objective is to apply this system for comparing and retrieving handwritten notes based on their visual characteristics. Specifically, the CBIR system's goal is to identify and return images that belong to the same category. In this context, the categories are defined by the different types of notes, primarily distinguished by variations in their sheet grid patterns. To achieve this purpose, we have explored and employed various backbones for conducting comparisons.

5.1 Datasets for Image Retrieval

We evaluated the RMAC-based retrieval system on three distinct datasets constructed using different methods: raw unprocessed scanned images of notes from various subjects and synthetic images generated for UNet network training.

- Dataset 1 ("db_0") consists of 120 unprocessed note images, with 20 images per class, from six different subjects.
- Dataset 2 ("db_1") comprises 120 images, including 60 without gridlines and 60 artificially generated images. These generated images are similar to the ones used for training the UNet network in image denoising. There are no associations between the images across the two classes.
- Dataset 3 ("db_2") includes 200 images, with 20 images without gridlines and 180 images generated from the previous set using nine different grid styles. These generated grid images are of the same type as the ones used for training the UNet network in image denoising. There are associations between the non-grid and generated grid images across the ten classes, with 20 images per class.

5.2 Methods

5.2.1 Image Retrieval Pipeline

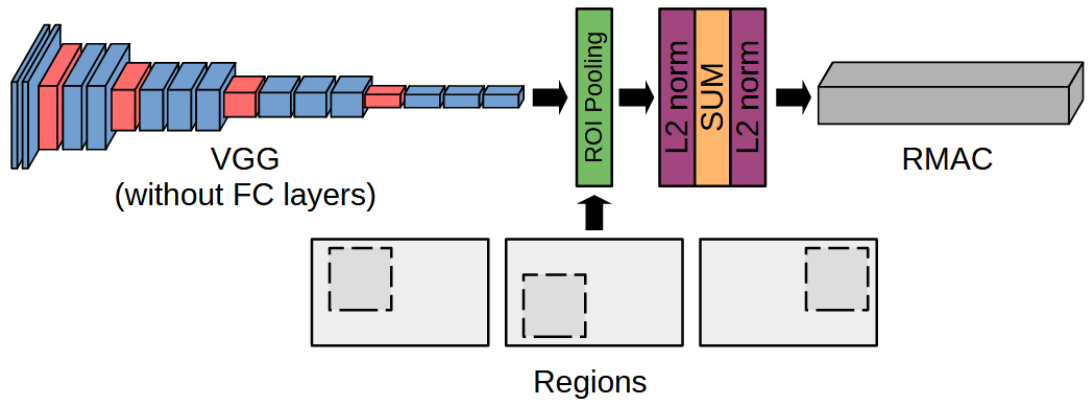


Figure 10: Pipeline of the RMAC vector extraction

1. Given a query image, it is first preprocessed and converted into a tensor.

2. The query image tensor is passed through the pre-trained CNN feature extractor to obtain the convolutional activation maps.

3. Regional maximum activation of convolutions (R-MAC) is applied on the activation maps to obtain a regional feature vector for each image region at multiple scales.

4. The regional feature vectors are l_2-normalized, summed and l_2-normalized again to obtain a single global image descriptor vector.

5. The query descriptor is compared against the precomputed database descriptors using cosine similarity to retrieve the top k most similar images.

5.2.2 Feature Extraction

Preprocessing and CNN Activations. Given an input image I of size $W_I \times H_I$, the image is passed through a pre-trained CNN. The fully connected layers of the network are discarded, and the activations of a convolution layer are considered. These activations form a 3D tensor of dimensions $W \times H \times K$, where K represents the number of feature channels, and W and H are the width and height of the feature maps, respectively [13] [14].

5.2.3 Maximum Activation of Convolutions (MAC)

The MAC representation calculates the maximum activation across each feature channel. This results in a vector of length K. Each component f_i in the feature vector corresponds to the maximum activation of the i-th feature channel [13] [14]. The MAC representation is given by:

$$f = [f_1, ..., f_k, ..., f_K]^T, \quad \text{with} \quad f_k = \max_{x \in X_k} x \tag{7}$$

5.2.4 Regional Maximum Activation of Convolutions (R-MAC).

To capture regional information, the 3D tensor is divided into R square regions sampled at different scales, with approximately 40% overlap between consecutives regions. The region size at the largest scale ($l = 1$) is set as large as possible, i.e. its height and with are both equal to $\min(W, H)$ of the 3D tensor Fig.[11]. At every other scale l, the width and height of each region is $2\min(W, H)/(l + 1)$ Fig.[11]. The feature vector associated with each region can be expressed as

$$f_{R_i} = [f_{R_i,1}, ..., f_{R_i,k}, ..., f_{R_i,K}]^T, \quad \text{with} \quad f_{R_i,k} = \max_{x \in R_{i,k}} x \tag{8}$$

The computed feature vectors are l_2-normalized, then summed into a single vector and l_2-normalized again.

$$F = \sum_{i=1}^{N} f_{R_i} = [\sum_{i=1}^{N} f_{R_i,1}, ..., \sum_{i=1}^{N} f_{R_i,k}, ..., \sum_{i=1}^{N} f_{R_i,K}]^T \tag{9}$$

The final dimension of F is equal to the number of feature channels K.

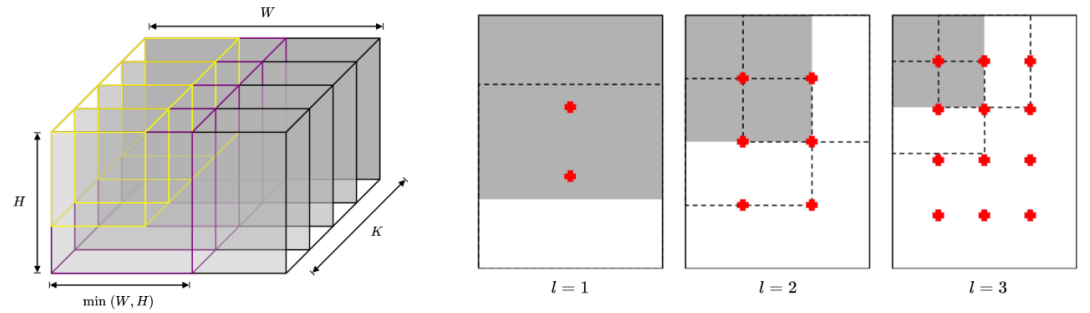


Figure 11: Left: 3D tensor output from the last CNN convolutional layer with dimension $W \times H \times K$. In purple, a sampled region of size $\min(W, H)$. In yellow, a region sampled at a smaller scale. Right: Sample regions extracted at 3 different scales (l = 1...3). We show the top-left region of each scale (gray colored region) and its neighboring regions towards each direction (dashed borders). We depict the centers of all regions with a cross.

5.2.5 Hyperparameters

- *Backbone CNN*: This refers to the pre-trained convolutional neural network used to extract features from the input image. The choice of backbone CNN can affect the performance of the image retrieval system.
- *Overlap between regions*: This is the amount of overlap between consecutive regions sampled from the 3D tensor. In this case, the overlap is set to approximately 40%.
- *Number of scales*: This refers to the number of scales at which the 3D tensor is sampled.

5.3 Esperiments and Results

We evaluate our image retrieval system primarily using the mean Average Precision (mAP) metric [12]. For a single query, the Average Precision (AP) is calculated as:

$$AP@k = \frac{1}{k}\sum_{i=1}^{k}\delta(y_i, y_q) \text{ where } \delta(y_i, y_q) = \begin{cases} 1 & \text{if } y_i = y_q \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Where:

- k is the number of images retrieved;
- y_i is the label of the i-th image retrieved;
- y_q is the label of the query image;

Thus, the average precision (AP) for a single query is the average of the δ relevance scores on the first k images retrieved. To calculate the mean average precision (mAP) on N query:

$$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N} AP_i@k \tag{11}$$

For the experiments, we evaluated several backbone architectures including: VGG16, VGG19, DenseNet, UNet trained on image denoising, and 3 UNet models with Kaiming initialization

and random weights. For VGG16, VGG19 and DenseNet, we used the last feature map before the fully connected layers. For UNet, we used the bottleneck feature from the encoder before the decoder.

The experiments were conducted on the three datasets presented earlier, using a 40% overlap ratio and 6 sampling scales. Each image in a dataset serves as a query for the CBIR system, and the AP is computed for each query image. The APs across all query images are averaged to obtain the overall mAP for that backbone architecture on the specific dataset.

We report the mAP@K for varying values of K - 3, 5, 10 and 20. Analyzing mAP over different top - K retrievals provides a more comprehensive view of the system's performance. The results on the different datasets are shown below:

Table 4: mAP on db_0

Model	k = 3	k = 5	k = 10	k = 20
VGG16	0.997	0.997	0.993	0.951
VGG19	0.997	0.998	0.994	0.930
DenseNet	1.000	1.000	0.998	0.923
Trained UNet	0.983	0.975	0.864	0.680
Kaiming UNet 0	1.000	0.995	0.989	0.923
Kaiming UNet 1	1.000	1.000	0.998	0.945
Kaiming UNet 2	1.000	0.998	0.996	0.935

Table 5: mAP on db_1

Model	k = 3	k = 5	k = 10	k = 20
VGG16	0.972	0.955	0.927	0.894
VGG19	0.950	0.937	0.905	0.876
DenseNet	0.972	0.958	0.932	0.894
Trained UNet	0.908	0.858	0.806	0.758
Kaiming UNet 0	0.939	0.903	0.854	0.768
Kaiming UNet 1	0.939	0.912	0.864	0.785
Kaiming UNet 2	0.906	0.868	0.837	0.740

On db_1, the pretrained models again achieve the best mAP scores. The UNet models have significantly lower mAP, but perform better than the trained UNet. Their performance stays relatively consistent even as K increases.

On db_2, the pretrained CNNs again outperform the UNet models significantly. However, their mAP scores are also lower on this dataset compared to db_0 and db_1. The UNet models achieve very poor performance on db_2. Their mAP scores are consistently low across different values of K.

Overall, the pretrained CNN models of VGG16, VGG19 and DenseNet consistently achieve the best performance on all 3 datasets. Their rich hierarchical features allow extracting powerful representations for effective image retrieval. The UNet models struggle to match their performance, particularly on db_2 which contains more complex synthetic images. The trained UNet performs the worst since it overfits to the denoising task.

Table 6: mAP on db_2

Model	k = 3	k = 5	k = 10	k = 20
VGG16	0.675	0.636	0.604	0.548
VGG19	0.717	0.711	0.674	0.594
DenseNet	0.730	0.724	0.677	0.613
Trained UNet	0.467	0.360	0.317	0.302
Kaiming UNet 0	0.467	0.360	0.354	0.348
Kaiming UNet 1	0.467	0.370	0.371	0.375
Kaiming UNet 2	0.467	0.360	0.347	0.336

6 Conclusion

This paper explores image denoising for handwritten notes using a UNet architecture. Compared to other techniques like Fourier transforms, UNet is more effective because it learns how to remove gridlines and noise from images. The paper presents a good pipeline for generating synthetic training data and preprocessing real images. This is a promising approach for preparing handwritten data for optical character recognition and math symbol detection.

Some key results from the paper:

- UNet model outperforms Fourier transform method for denoising gridlines in images.
- Data augmentation and use of SGD optimizer led to improved model training and performance.
- Retrieval system based on RMAC vectors can identify images requiring denoising.

Overall this work provides a solid foundation for removing background noise from handwritten notes to improve downstream tasks like OCR.

7 Future Developments

Our proposed work could continue in the following ways:

- The training dataset could be further augmented with more variations to better match real images.
- Instead of training UNet from scratch, fine tune a pretrained model could boost performance.
- Alternative learning rate schedulers like cyclical or one policies could be explored.
- More advanced metrics beyond MSE/SSIM would better evaluate similarity of denoised images to ground truth.
- Using Vision Transformer (ViT) or DINO for image embeddings may improve the retrieval system.
- Retrieval system could be extended to find similarity between crops rather than full images.

In summary, this work establishes a good starting point for image denoising. Several promising directions exist to enhance the data, model training, evaluation metrics and retrieval techniques. Pursuing these improvements could strengthen the overall pipeline and results.

A Appendix: Removing the grid of squares using FFT

In order to remove the grid of squares the Fourier transform was useful to analyze the frequency components of an image, and since the horizontal and vertical lines correspond to a low frequency components of the images spectra, in fourier domain we are able to remove this components with other classical image processing techniques. Then the morphological trasformation has been used to remove the remaining noise after the images was filtered in fourier domain.

A.0.1 Fourier Transform

The first step in our pipeline is convert the color image to a grayscale. This conversion should simplifies the subsequent processing steps and reduces the computational cost. We apply a 2D Fourier transform to the grayscale image using NumPy's fft2 function. The resulting complex-valued array contains information about the frequency components of the image to apply a 2D Fourier transform to the grayscale image of the handwritten math note.

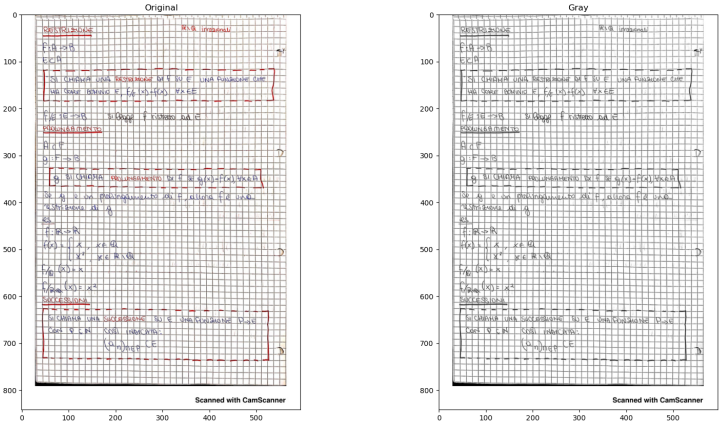


Figure 12: Sample image and image converted to grayscale

Next to extract the relevant information from the Fourier transform, we compute the magnitude and phase spectra. The magnitude spectrum represents the distribution of energy in the frequency domain and is computed by taking the absolute value of the Fourier transform and shifting the zero-frequency component to the center of the spectrum. The phase spectrum represents the phase of the frequency components and is computed by taking the angle of the Fourier transform and shifting the zero-frequency component to the center of the spectrum. Both spectra are visualized using a logarithmic scale to enhance the contrast between different frequency components and helps to identify the foreground (formula) in the image.

A.0.2 Thresholding

To separate the foreground (formula) from the background in the image, we apply a threshold to the magnitude spectrum. The threshold value is computed as a fraction of the maximum

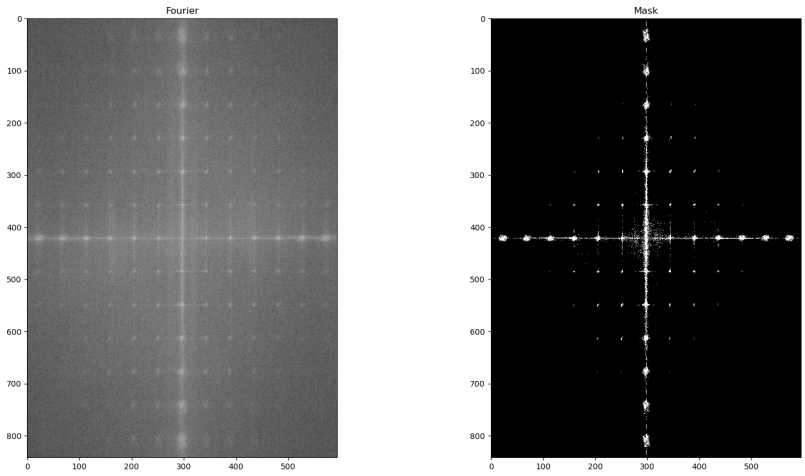


Figure 13: Fourier transform and mask

magnitude value in the spectrum. Pixels with magnitude values above the threshold are considered part of the foreground and are set to 1 in a binary mask. Pixels with magnitude values below the threshold are considered part of the background and are set to 0 in the binary mask. The binary mask is created from the thresholded magnitude spectrum. Then we find the image pixel indices that match the foreground (formula) by selecting the pixels where the binary mask is true. The logarithmic magnitude values of the foreground pixels are replaced with the average value of the background pixels to reduce noise and improve contrast between foreground and background. An exponential transform is then applied to the modified logarithmic magnitude spectrum and combined with the phase spectrum to obtain a new complex-valued Fourier transform. after which an inverse Fourier transform is applied to the shifted Fourier transform to obtain a new image. Subtract the resulting image from 255 to obtain a binary white-on-black image of the formulas.

A.0.3 Morphological Operations

To further refine the binary image of the formulas, we apply morphological operations to remove small false-positive detections and improve the connectivity of the equation. A combination of erosion and dilation operation were used to remove small isolated regions and fill gaps in the formulas. In particular in first intance we use erosion followed by dilation to remove the white pixel of noise from the negative.

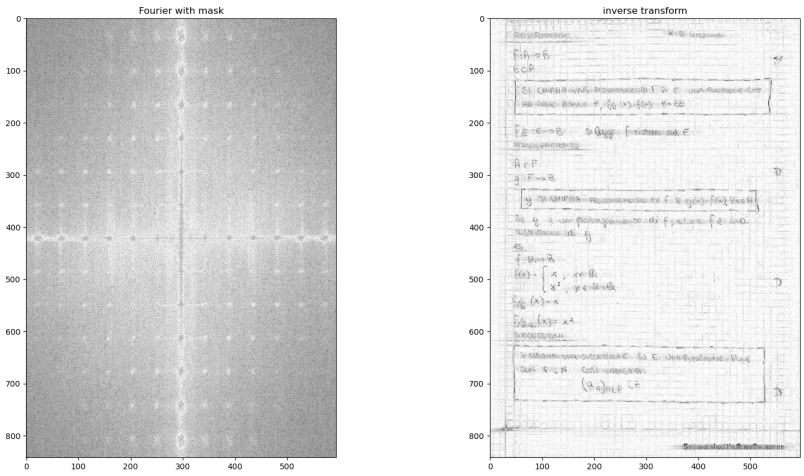


Figure 14: Here the mask is applied to the fourier transform and then the anti transform is done

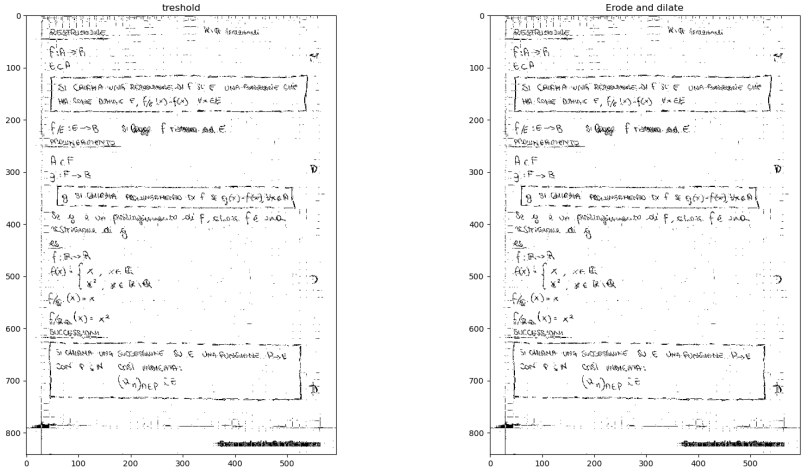
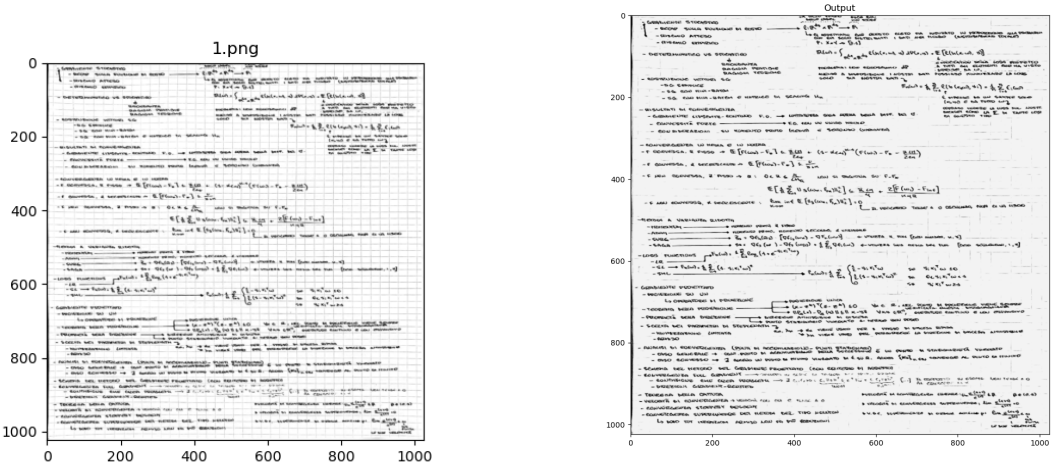


Figure 15: Threshold and then morphological operations are applied

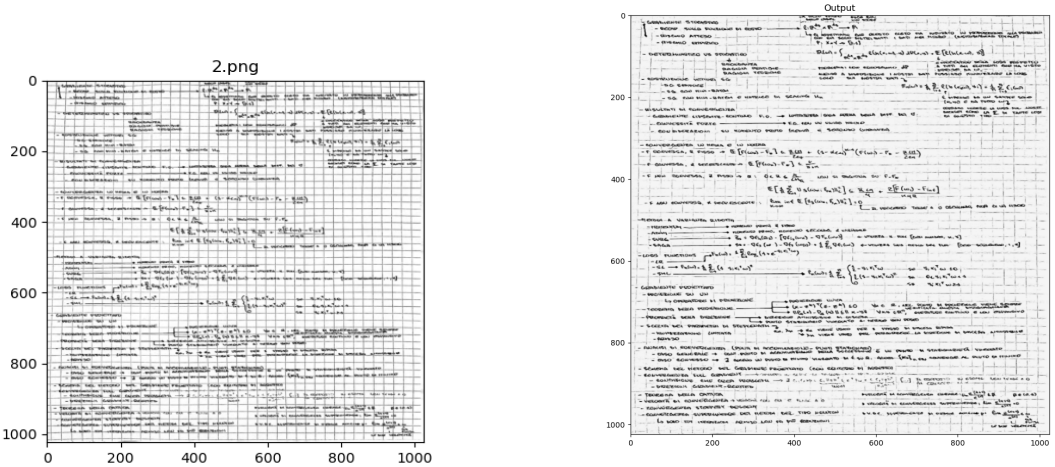
B Appendix: Denoising Experimental Results

In this section is presented additional experimental results of the denoising with UNet, using different version of the model, trained with different optimizators and different version of the dataset.



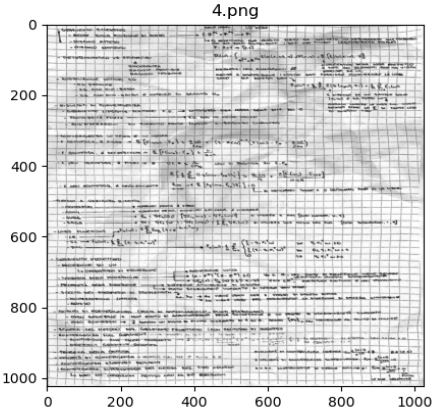
(a) Image with grid (b) Inferenced

Figure 16: Inference on Adam without augmentation



(a) Image with grid (b) Inferenced

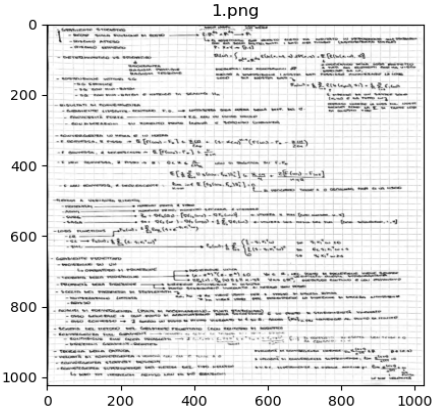
Figure 17: Inference on Adam with augmentation



(a) Image with grid

(b) Inferenced

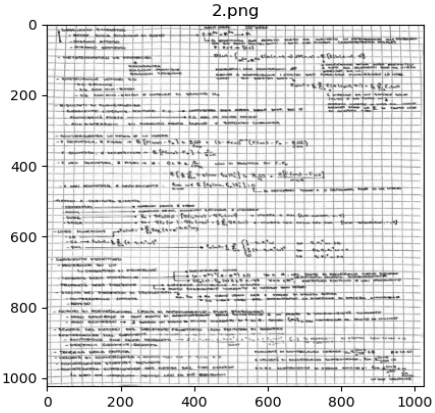
Figure 18: Inference on SGD with augmentation



(a) Image with grid

(b) Inferenced

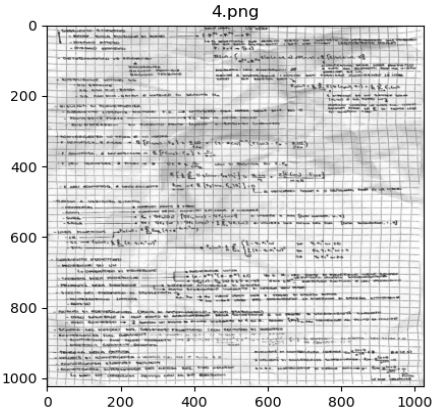
Figure 19: Inference on Adam without augmentation



(a) Image with grid

(b) Inferenced

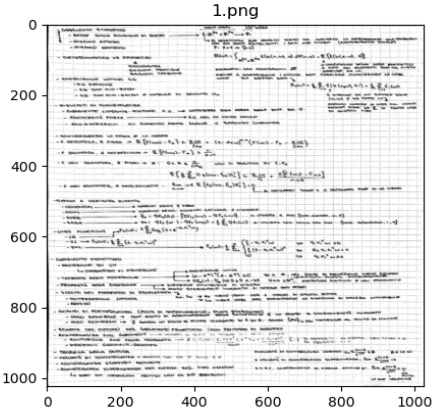
Figure 20: Inference on Adam with augmentation



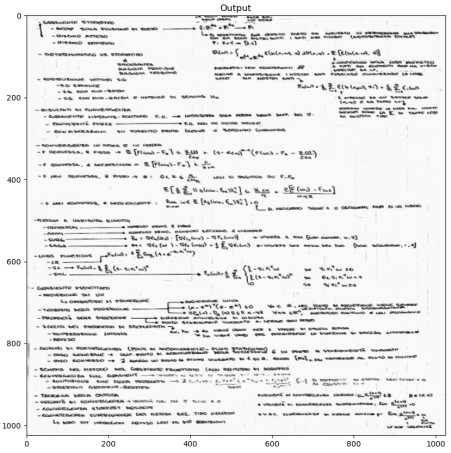
(a) Image with grid

(b) Inferenced

Figure 21: Inference on SGD with augmentation

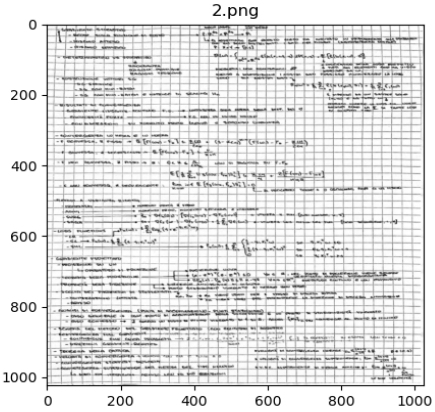


(a) Image with grid

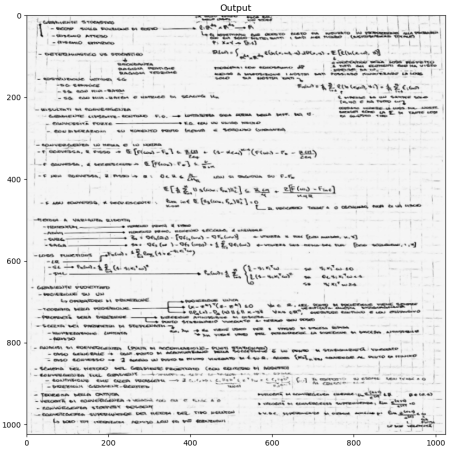


(b) Inferenced

Figure 22: Inference on Adam without augmentation

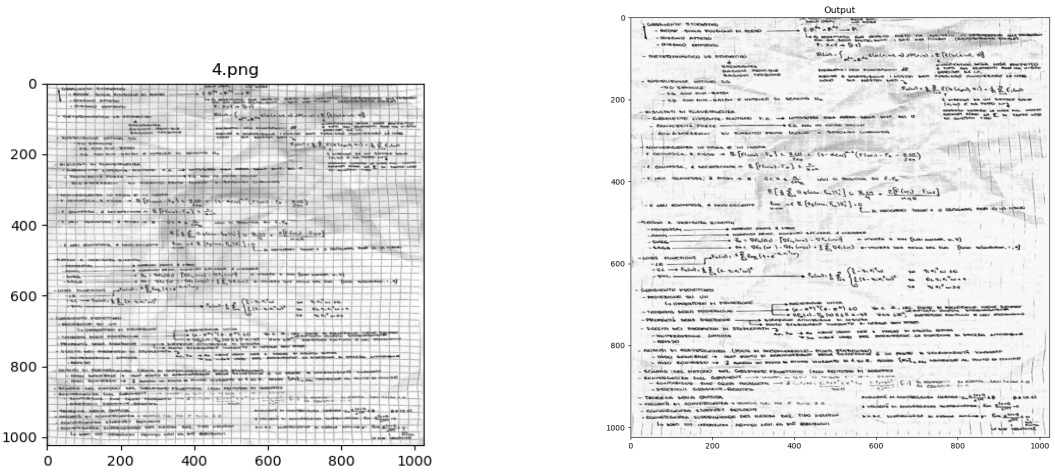


(a) Image with grid



(b) Inferenced

Figure 23: Inference on Adam with augmentation



(a) Image with grid

(b) Inferenced

Figure 24: Inference on SGD with augmentation

C Appendix: CBIR Experimental Results

This section shows examples of using the image retrieval system. The first example concerns the use of the system on db_0 with VGG16 pretrained, and with K = 10. The second example concerns the use of the system on db_2 with VGG16 pretrained, and with K = 20.

Figure 25: Rietrieval on db_0 with k = 10 using VGG16

Figure 26: Rietrieval on db_2 with k = 20 using VGG16

References

[1] Paavani Dua Department of Electrical Engineering Stanford University *Image Denoising Using a U-net* http://stanford.edu/class/ee367/Winter2019/dua$_r eport.pdf$

[2] Jiajin Zhang, Hanqing Chao, Xuanang Xu, Chuang Niu, Ge Wang, Pingkun Yan
Task-Oriented Low-Dose CT Image Denoising https://arxiv.org/abs/2103.13557

[3] Qingsong Yang, Pingkun Yan, Mannudeep K. Kalra, Ge Wang
CT Image Denoising with Perceptive Deep Neural Networks https://arxiv.org/abs/2103.13557

[4] Sharath Solomon
Image Denoising using Deep Learning https://medium.com/analytics-vidhya/image-denoising-using-deep-learning-dc2b19a3fd54

[5] *Morphological Transformations*, https://docs.opencv.org/4.x/d9/d61/tutorial$_p y_m orphological_o ps.html$

[6] *Geometric Transformations of Images* https://docs.opencv.org/4.x/da/d6e/tutorial$_p y_g eometric_t ransformations$

[7] Removing lines from an image a notebook for digit detection python,
https://stackoverflow.com/questions/29360025/removing-lines-from-an-image-a-notebook-for-digit-detection-python

[8] Image Thresholding
https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html

[9] Canny Edge Detection
https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html

[10] How to Build a Kick-Ass Mobile Document Scanner in Just 5 Minutes
https://pyimagesearch.com/2014/09/01/build-kick-ass-mobile-document-scanner-just-5-minutes/

[11] Extract horizontal and vertical lines by using morphological operations,
https://docs.opencv.org/4.x/dd/dd7/tutorial_morph_lines_detection.html

[12] Konstantin Schall, Kai Uwe Barthel, Nico Hezel, Klaus Jung *GPR1200: A Benchmark for General-Purpose Content-Based Image Retrieval* https://arxiv.org/abs/2111.13122

[13] Giorgos Tolias, Ronan Sicre, Hervé Jégou *Particular object retrieval with integral max-pooling of CNN activations* https://arxiv.org/abs/1511.05879

[14] Yang Li, Yulong Xu, Jiabao Wang, Zhuang Miao, Yafei Zhang *MS-RMAC: Multiscale Regional Maximum Activation of Convolutions for Image Retrieval* IEEE Signal Processing Letters, Vol. PP, pp. 1-1, Feb 2017 https://doi.org/10.1109/LSP.2017.2665522

List of Figures