Utrecht University

Department of Information and Computing Science

Game and Media Technology

Re-ordering The Sequence of Errera Sketchbook

First examiner:

Remco Veltkamp

Second examiner:

Almila Akdag

Candidate:

Zuchi Zheng

In cooperation with:

Daantje Meuwissen

August 30, 2024

Abstract

The thesis focuses on the re-ordering of the Errera Sketchbook, which is significant for the study of early modern art practice. Sketchbooks serve as a crucial tool for artists, helping them train, collect patterns, record travel, and experiment. The Errera Sketchbook's original order has been disrupted, making it challenging for art historians to study the progression of ideas and techniques within it. This research employs both traditional art historical analysis and advanced image processing techniques, including YOLOv8 for object detection, to restore the sketchbook to its original sequence. The project aims to offer a novel approach to the preservation and study of historical artworks by combining art history with AI technology.

Contents

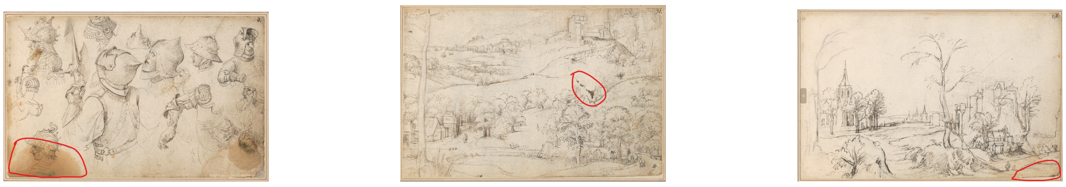
1. Introduction

1.1 Motivation

The importance of sketchbooks in the study of early modern art practice is widely recognized. Sketchbooks are a tool for artists to trains, collects patterns, records travel, and experiments. The sketchbook serves as an artist's intimate repository of creative endeavors, intended exclusively for the artist or their atelier. Consequently, scrutinizing the sketchbook can be construed as an inquiry into private artistic praxis. Recent research[1] has indicated that many of these sketchbooks were not originally bound as books, but were made up of loose manual pages. It was later that the sketchbooks were casually organized into book form. We're going to look at a 16th-century Dutch sketchbook called the Errera Sketchbook, made between 1520 and 1530. The book originated in Antwerp, probably from the studio of artist Joachim Patinil (c. 1480-1524), and was bound in a leather binding from the 19th century[1]. The sketch book consists of 85 folios drawn on recto and verso (folios refers to the fact that centuries ago, artists would make books by folding sheets of paper in half and binding them together. Each sheet of paper was folded in half to create two folios, verso is the "left" or "back" side and recto is the "right" or "front" when the content is written on folios), but their current structure, organization, and sequence are not original, as evidenced by (19th century) drawing numbers. Hence, scholars aim to better comprehend history and the experiences of the author by reconstructing the original sequence of the sketchbook. Currently, the sketchbook has been in the Royal Museum of Fine Arts in Brussels since 1928.

1.2 Dataset

By high resolution scanning, we obtained a dataset of around 180 images from Errera Sketchbook. These pictures contain not only drawings, but also material information, as shown in figure 1.1. For example, water-stains, binding holes, remnants of ink or other drawing materials from one folio on another, and rounded edges. This information will be used as evidence to find the original sequence of the sketchbook such as the binding holes of two folios in the same position, which may indicate that they are from the same paper.



(a) Water-strains **(b)** Ink remnants **(c)** Rounded edge

Figure 1.1: Material information

Ultimately, after scanning the sketchbook, the dataset comprised a total of 183 images. However, not all of these images were useful; some were merely blank pages with no relevant information, as show in figure 1.2 .

Figure 1.2: Page E0a-B1, there is no information used to determine the location of this page

Additionally, there were images containing content that was not pertinent to our study, such as the cover of the sketchbook, as show in figure 1.3.

Figure 1.3: Ecover, no need to define its position in the sketchbook

After selection, the following images were removed from the dataset: E0a-B1, E0b-B1, E0c-B1, E0d-B1, E33v-B6, E37v-B5, E39v-B5, E49v-B7, E53v-B6, Ecover1-B9, Ecover2-B9, Ecover3-B10, Ecover4-B10, Espread1-B10, Espread2-B10, and Espread3-B10. Further details can be found in the appendix 7.1.

2. Related work

In this section, related work and research will be introduced. Although the research of this project is to design artificial intelligence to complete the re-ordering of the sketchbook, most of the knowledge involved in this process is related to computer science, we also need to do some research on art and history. Because it allows us to understand more about the content and details in the image, which information is useful.

2.1 Art history

Through the study of art history, we can have a better understanding of these pictures and help us capture more details and features. As I mentioned above, Errera sketch book not only records the master's training, collect patterns, record travel, and experiments at that time [1]. The sketchbook mainly records the author's travel in the Low Countries at that time, drawing buildings, landscapes, statues and murals [2]. At the same time, they can also establish connections with other artists, so we can understand the history better. For example, Cesare da Sesto's sketchbook kept in the Morgan Library in New York, recorded his travel experience in Rome during the Renaissance. Although Cesare is not well known in art history, by analyzing his painting style, scholars have identified him and three major artists: The careers of Da Vinci (1452-1519), Raphael (1483-1520) and Michelangelo (1475-1564) have a brief overlap [3]. In general, these sketchbooks contain a lot of information, which is not only the legacy of art, but also the evidence we can rely on to understand the history.

Through the study and understanding of art history, we found that there are many features in these sketchbooks that we can use to re-order the sketchbooks.

2.1.1 Paper material

First, we pay attention to the material of the paper. Because in that period, different countries, royal used, civic used, they all had their own specifications[3]. As shown in figure 2.1, we can find in the sketchbook that these folios obviously use different materials of paper.

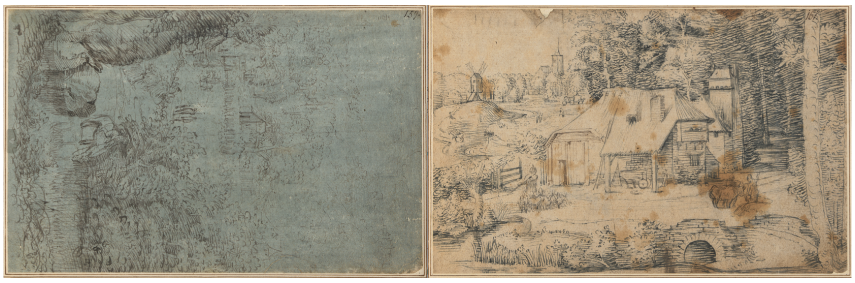
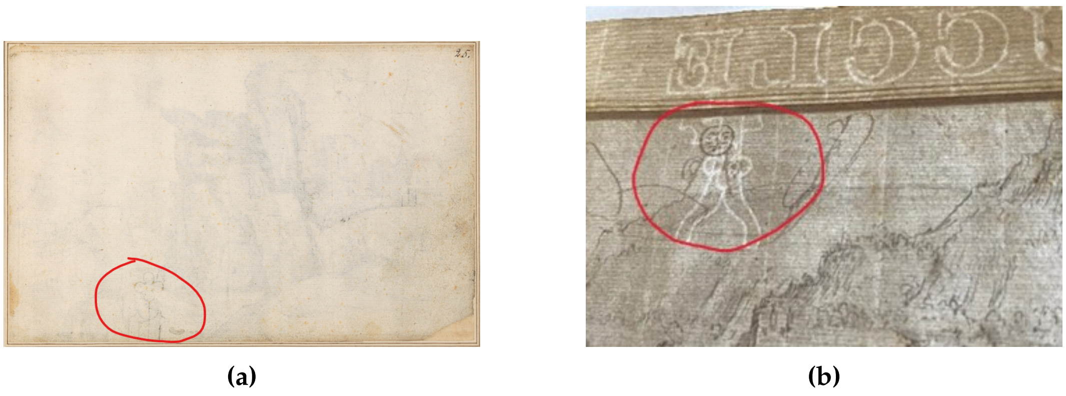


Figure 2.1: Different material

2.1.2 Watermark

Secondly, watermark is also the evidence that researchers refer to. They indicate that the watermark help determine that the papers were made by the same mold, thus inferring that the papers may have come from the same time and place [1], [3], [4], as shown in figure 2.2. For example, the authors speculated that the papers might be French paper rather than central Italian paper by analyzing the types of watermarks "drawing".



(a) **(b)**

Figure 2.2: Different watermarks

2.1.3 Chain lines

First of all, the process of making the paper at the time of sketchbook creation needs to be elaborated. Before the mid-18th century, paper was formed by dipping sieve-like moulds into pulp immersion tanks and removing them in as uniform a manner as possible to coat the moulds with pulp while gradually draining the water. This mold consists of a grid with widely spaced metal lines (chain lines) and, vertically, more densely spaced metal lines (horizontal lines) and watermark lines. This mold forms a grid-like structure on the paper, the so-called horizontal and chain line patterns, along with a watermark identifying the paper factory or country of origin. The horizontal and chain line patterns as well as the watermark part are more transparent than the rest of the paper, and are particularly noticeable when viewed under transmitted light [4], as shown in the figure 2.3.

(a) Chain line on Errera sketchbook **(b)** Chain line on Cesare's sketchbook

Figure 2.3: Chain lines

Therefore, we can rely on the structure of the chain lines and the distance between them to identify the origin of the paper. As we mentioned earlier, the sketchbook records the drawings of the author in different places and at different times, so when the author stays in a certain place, he is likely to use the paper of the same origin. As a result, we can then classify the papers with the same chain lines in the sketchbook into the same period. The dates of these works are about the same and there will be continuity.

2.1.4 Contents of the painting

By observing the content of the drawings, as shown in figure 2.4, the content in E51v and E53 is continuous. The content extends from one sheet of paper to another, demonstrating that they originate from the same sheet of paper before being bound. Based on this feature, the two sheets of paper can be matched. Thus, their sequence should be consecutive. However, this feature is not suitable for computational techniques.

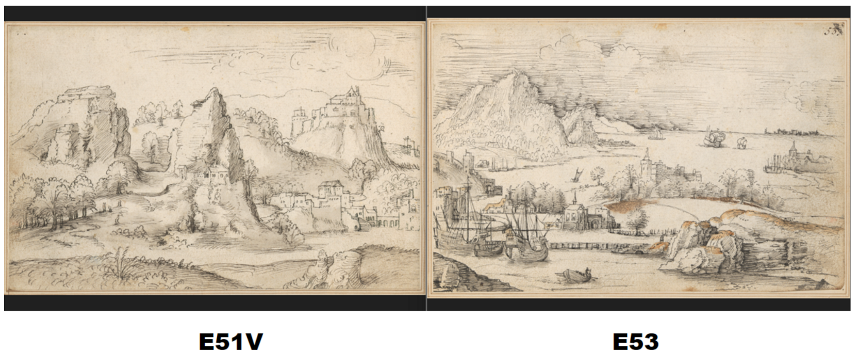
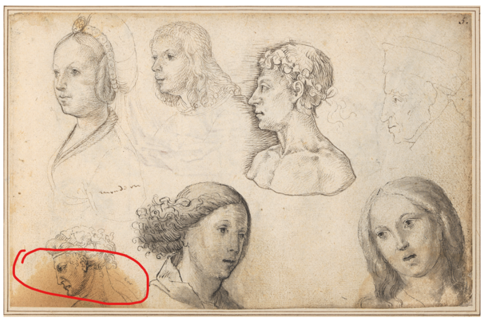


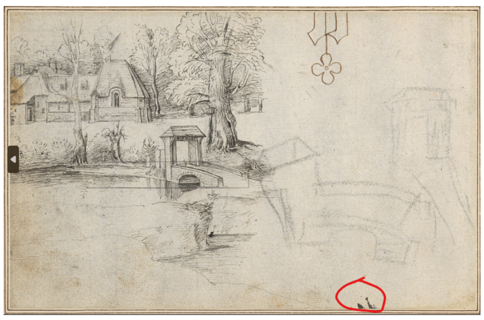
Figure 2.4: Continuous

2.1.5 Water stains and ink remnants

Water stains and ink remnants in figure 2.5 have permeability. When a drop falls on the picture, the stain will appear on the adjacent folios through the folios [1], [4].



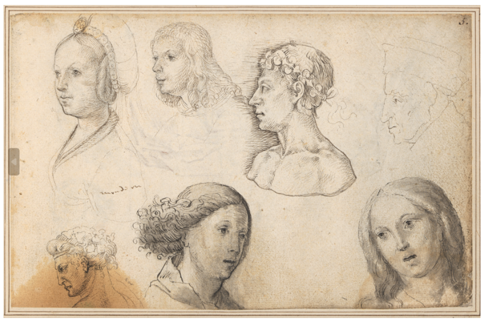
(a) Water stains



(b) Ink remnants

Figure 2.5

For example, in figure 2.6, by comparing right image and left image, we can find that they have similar water stains, but they appear in opposite positions. From this feature, we can infer that the two drafts should belong to the front and back sides of the same paper.


(a) E5

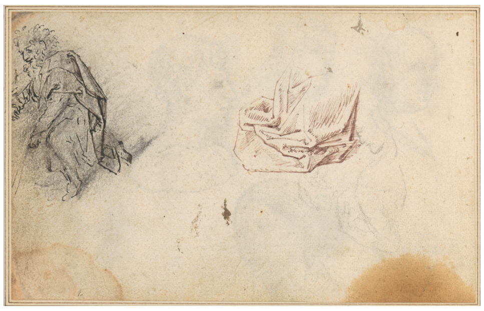
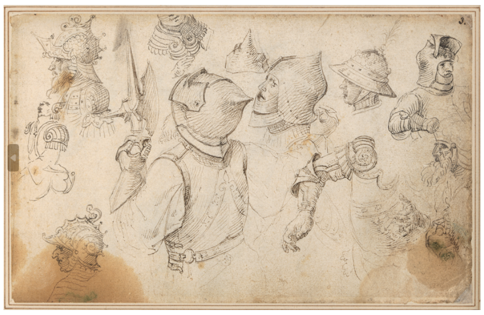

(b) E5V

Figure 2.6

Also, in Figure 2.7 comparing right with left, we can see that they have similar stains and are in the same position. However, the area of the stain in right is smaller than that in left. This is obviously due to the fact that the stain falls on left and permeates right. That means they're adjacent.


(a) E3

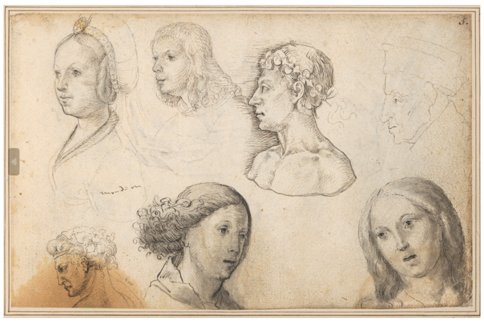

(b) E5

Figure 2.7

2.1.6 Conclusions

In summary, the above describes the art history research and paper material analysis involved in the research of sketchbook re-ordering. However, not all features are helpful in the original ordering of the sketch book. For example, chain-lines and water marks only help us determine the paper′ manufacturers, classifying the paper by it does not help us determine its order, and the feature of continues drawings is hard to learn by AI. Therefore, in the project, only water stains and ink remnants help to re-order the sketchbook.

2.2 HSV color space

In the research, additional images needed to be added because of lack of the data. For the style consistent of image, the hue, saturation, and value of brightness need to be adjusted in HSV color space as shown in the figure 2.8.

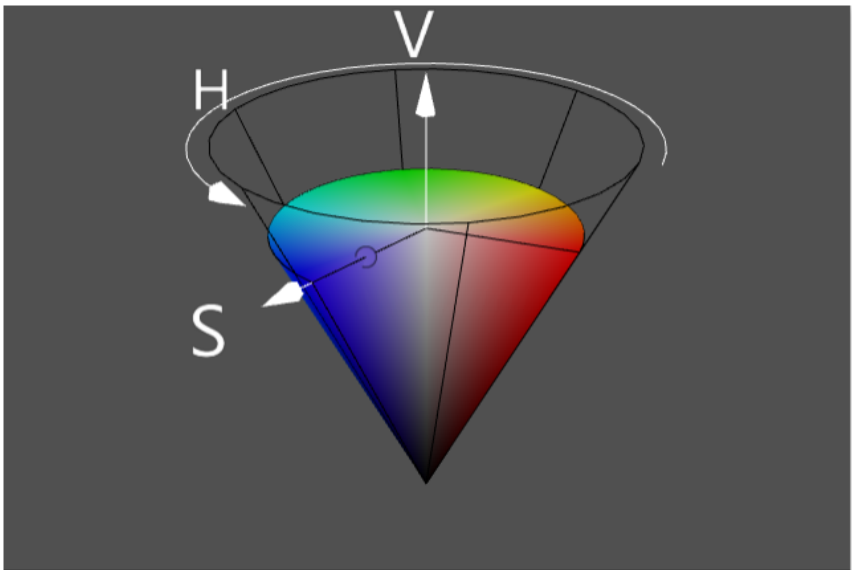


Figure 2.8: HSV color space: H is hue, S is saturation, V is value of brightness [5].

By compared with the RGB, HSV is more consistent with the perception of human visual system [6].

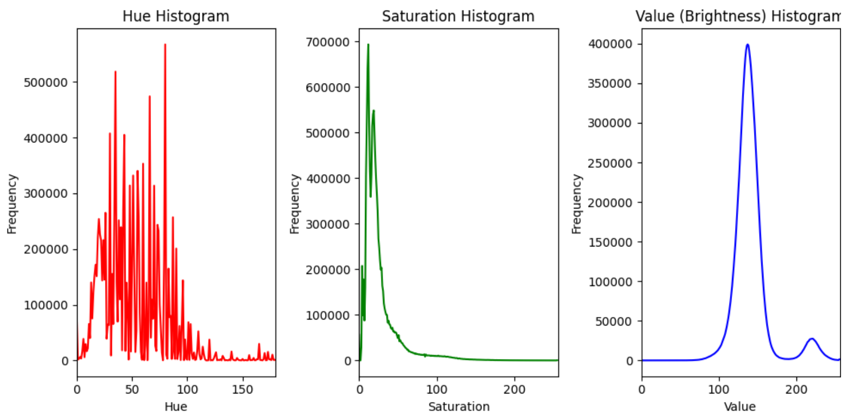


Figure 2.9: The average of all images in the HSV color space

2.3 YOLOv8

In the field of image processing, traditional algorithms are constrained by high-dimensional data space and challenges in feature extraction. These algorithms require manual design and feature extraction, but often fail to capture local structures and complex patterns in images, thus struggling to effectively address problems such as image similarity or recognition. In contrast, Convolutional Neural Networks (CNNs) leverage mathematical principles such as local connectivity, weight sharing, and multi-level feature extraction to automatically learn feature representations of images. This means that CNNs can learn task-specific feature representations from raw image data without manual intervention. Furthermore, the convolutional layers of CNNs exhibit translation invariance, allowing them to recognize objects even when they undergo translation within the image. By progressively extracting features through multiple layers of convolutional and pooling operations, CNNs can gradually capture abstract features in images, from low-level to high-level, thereby enhancing the model[7].

The sketchbook database is then constructed to train the CNN in identifying water stains and ink stains within the images, thereby extracting features. Consequently, a CNN model is required that takes an image as input and outputs labels, sizes, positions, and confidence levels of the feature information contained in the image. Notably, bounding boxes can be utilized to represent feature sizes and positions. The network architecture of YOLOv8 [8] is a reference for this purpose.

2.3.1 YOLOv8 workflow

The YoloV8 network divides the input image into an $S \times S$ grid and performs object detection in each cell, as illustrated in the figure 2.10.

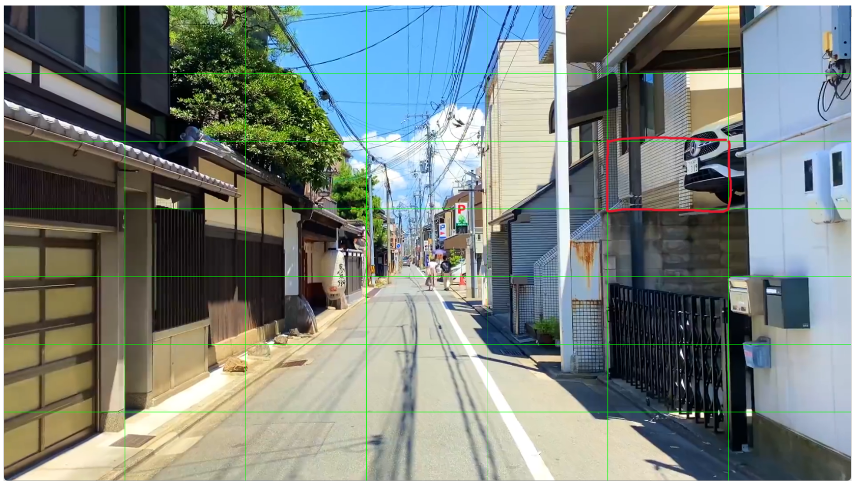


Figure 2.10: Grid view

In the highlighted red area, cells contain cars and each cell predicts B bounding boxes along with their corresponding confidence scores. The confidence score indicates the likelihood of a bounding box containing an object and its accuracy. The former is represented by $Pr\left(object\right)$, which is 0 if the bounding box does not include the target and 1 otherwise. The accuracy of the bounding box can be measured by calculating the IOU (intersection over union) ratio between the predicted b-box and the actual b-box, denoted as $IOU_{Predict}^{True}$. Therefore, the formula for confidence is as follows: $[x, y, w, h, c]$, where x and y represent the position of the prediction box; w and h denote its length and width; c represents its confidence score. For instance 2.11, consider a $7 \times 7 \times 13$ tensor that includes class probabilities in its first three elements. Since each cell predicts two bounding boxes, elements 4 and 5 correspond to their confidences, while elements 6-13 provide information about both sets of bounding boxes.

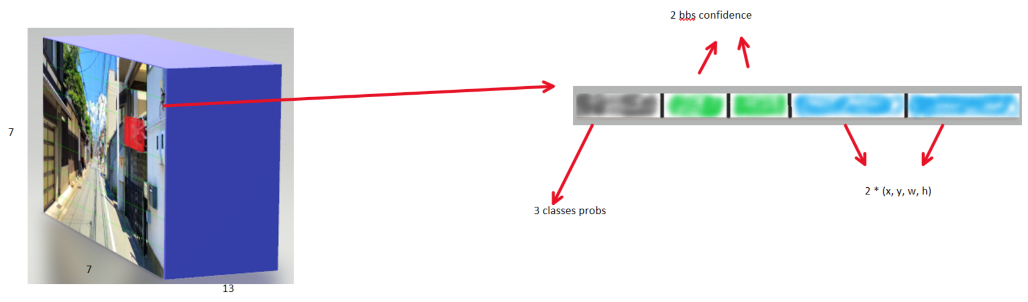


Figure 2.11: Output

However, in fact, when YoloV8 output the final result, it will have multiple bounding boxes for the same object. For this, it uses non-maximum suppression (NMS). First, all bounding boxes are sorted by their confidence in descending order. The IOU of the current bounding box with other bounding boxes is detected once, and if it is greater than the threshold, the other bounding boxes are removed. This step is repeated until we have a set of non-repeating bounding boxes as the final output.

$$IOU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$

2.3.2 YOLOv8 base structure

Here is a schematic diagram 2.12 of the basic structure of Yolov8[9], consisting of two main components. The first one is the backbone, responsible for feature extraction, and the second one is the head, responsible for predicting results.

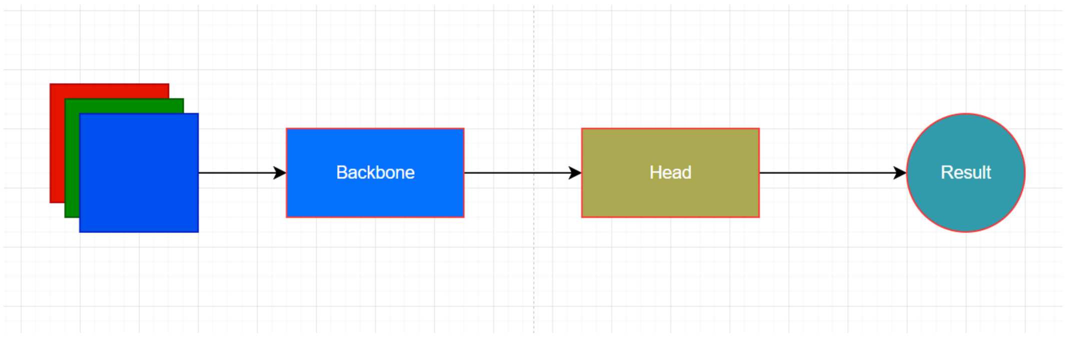


Figure 2.12: A summary of the structure

The detailed structure of the backbone and head is shown in the figure 2.13. In the backbone, there are mainly Conv blocks and C2f blocks, as well as an SPPF block (Spatial Pyramid Pooling Fast).

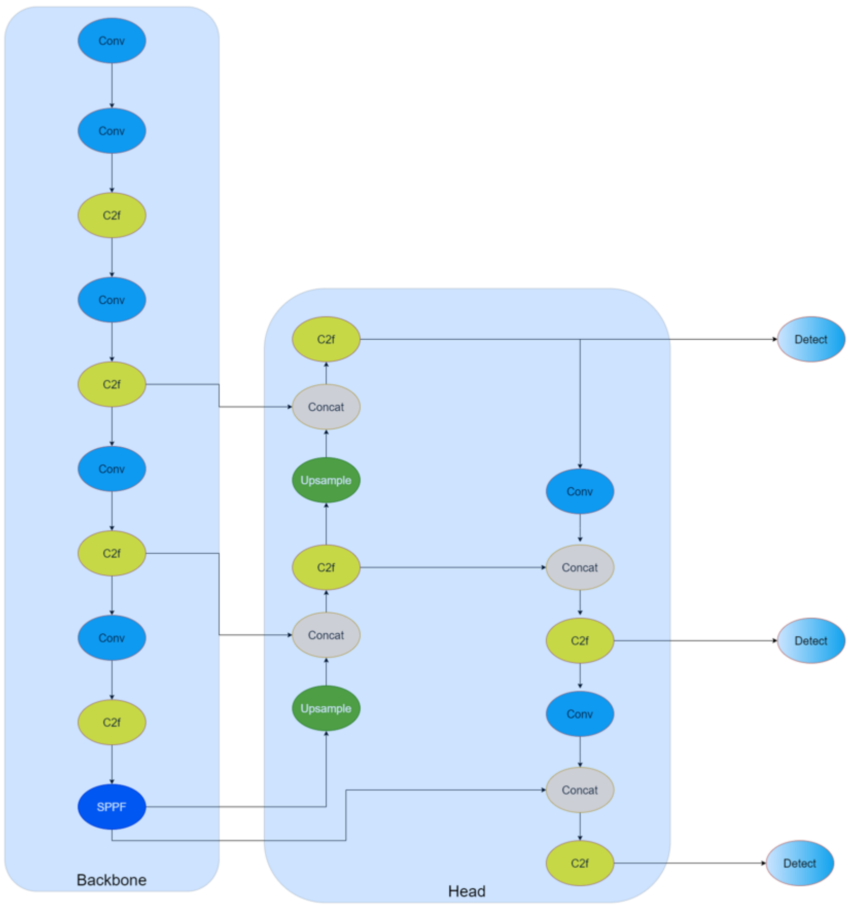


Figure 2.13: Main structure

Convolutional block (Conv block) 2.14 consists of a convolutional layer, followed by batch normalization (BatchNorm2d), and then the SiLU activation function (Sigmoid Linear Unit). The formula for SiLU is as follows:

$$SiLU\,(x) = x \cdot \sigma\,(x)$$

where $\sigma(x)$ represents the sigmoid function:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

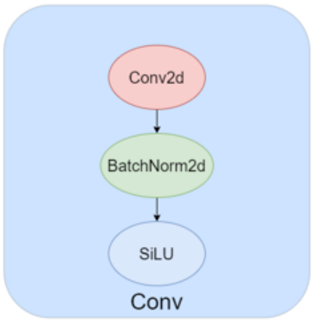
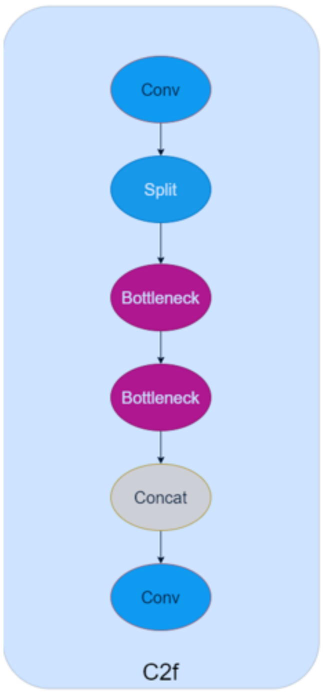
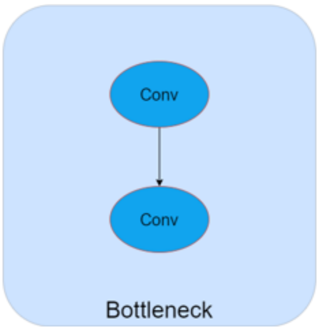


Figure 2.14: Conv

The structure of C2f (Conv to Fused) is shown in the figure 2.15a. It consists of an input layer and an output layer, both of which are Convolutional layers. It also includes n Bottleneck modules, each consisting of two Convolutional layers, as shown in the figure 2.15b. These modules are then linked to a Concatenation (Concat) layer, which is used to merge multiple tensors together to produce a higher-dimensional tensor, thus fusing features from different layers.



(a) C2f



(b) Bottleneck

Figure 2.15: C2f and Bottleneck

In the backbone, the output layer is SPPF (Spatial Pyramid Pooling Fast), which consists of two Convolutional layers, three pooling layers, and a Concatenation (Concat) layer, as shown in the figure 2.16. SPPF is responsible for enhancing the network's adaptability to images of different sizes.

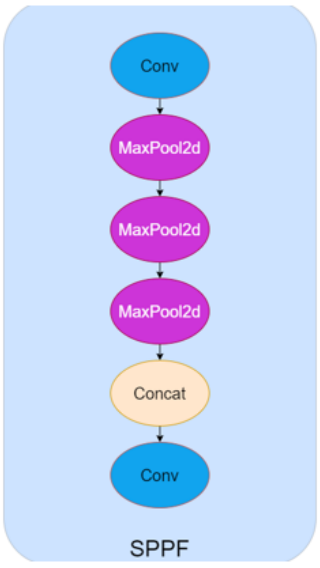


Figure 2.16: SPPF

In the Head section, it takes the output of SPPF as input. Additionally, it receives outputs from C2f of the Backbone at the first, second, and last Concat layers for fusion. In the second, third, and fourth C2f layers, the output is passed to the Detect module for predicting results at different scales. The structure of the Detect module consists of two branches: the first one predicts the information of bounding boxes, and the second one predicts the class of the targets, as shown in the figure 2.17.

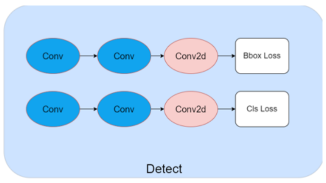


Figure 2.17: Detect

The full structure is in the appendix 7.1.

2.3.3 Loss function

Since the network needs to predict both bounding boxes and class, two different loss functions are used. The problem of bounding boxes is understood as regression loss, and the CIoU (complete Intersection over Union)[10] loss function is used. Label prediction is understood as classification loss, and the BCE (Binary Cross Entropy)[11] loss function is adopted. The formula of CIoU is as follows:

$$L_{CIoU} = IoU - \frac{d^2\left(center, center^{gt}\right)}{c^2} - \alpha v$$

α is the weight coefficient used to balance the influence of position and shape; c is diagonal length of bounding box; v is the ratio consistency measure between the predicted box and the ground truth box. IoU is intersection over union, A is predicted bounding box, and B is the ground true. The formula is:

$$IoU = \frac{|A \cap B|}{|A \cup B|}$$

$$\alpha = \frac{v}{(1 - IoU) + v}$$

$$v = \frac{4}{\pi^2} \left(arctan \frac{w^{gt}}{h^{gt}} - arctan \frac{w}{h} \right)^2$$

The loss function used for predicting categories adopts BCE, which is a commonly used loss function in binary classification problems. Here, y represents a binary label (0 or 1), and $p(y)$ represents the probability of outputting the label y. It is formulated as:

$$L_{BCE} = -[y \log(p(y)) + (1 - y) \log(1 - p(y))]$$

2.4 Convolutional block attention module(CBAM)

CBAM [12] is a lightweight and efficient attention mechanism designed to enhance the feature representation capability of convolutional neural networks (CNNs). It achieves this by sequentially applying channel attention and spatial attention, allowing the network to focus more effectively on the most important parts of the input features. The working principle of CBAM can be summarized in two steps: First, the channel attention module analyzes global information along the channel dimension to generate a channel attention map, which is used to reweight the importance of each channel.

Second, the channel-weighted feature map is fed into the spatial attention module, which analyzes the spatial dimension of the feature map to generate a spatial attention map, further optimizing the spatial distribution of features. Through these two steps, CBAM effectively enhances the network's representational power and inference performance, making it suitable for integration into various existing CNN architectures. CBAM's architecture is shown in the figure 2.18.

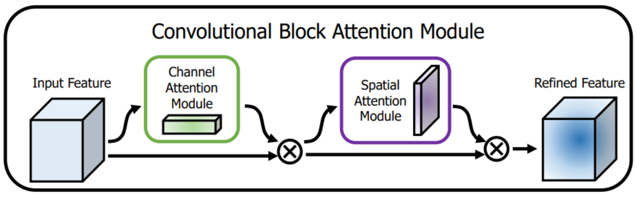


Figure 2.18: The overview of CBAM. The module has two sub-modules: channel attention and spatial attention [12].

2.4.1 Channel attention module

In channel attention module, Input Feature Map F is processed through average pooling and max pooling operations, generating two different feature descriptors:

$$F_{avg} = AvgPool(F), F_{max} = MaxPool(F)$$

These two descriptors are passed through shared fully connected layers to generate the channel attention map:

$$M_c(F) = \sigma(W_1(W_0(F_{avg})) + (W_1(W_0(F_{max})))$$

Where W_0 and W_1 are shared fully connected layer. σ is the *Sigmoid* activation function. Then, the channel attention map is multiplied with the input feature map F to create a weighted feature map:

$$F' = M_c(F) * F$$

2.4.2 Spatial attention module

The first part is similar to channel attention module, however, it get input feature map from F', and generate two different feature descriptors: F'_{avg} and F'_{max}. These two are concatenated and passed through a convolutional layer to generate the spatial attention map:

$$M_s(F') = \sigma(f^{7*7}(F'_{avg} \otimes F'_{max}))$$

where f^{7*7} means a convolution operation with a $7 * 7$ filter, and $\otimes$ represents the concatenation. The generated spatial attention map will be multiplied with the weighted feature map F', and produce the final output F'':

$$F'' = M_s(F') * F'$$

3. Basic Image Processing Approach

First, I tried the relatively simple solution, which is to solve the problem by comparing the similarities without using AI. The first is image preprocessing, we need to highlight the features and reduce the interference of drawing. Here, the image is first converted to gray-scale and then filtered so that only a certain range of images is shown. In order to observe the performance of various features in different intervals. As shown in the gray-scale histogram 3.1, the red line is lower-bound and the green line is upper-bound. After processing, only the pixels within the interval will be retained in the image, while those outside the interval will be replaced with white.

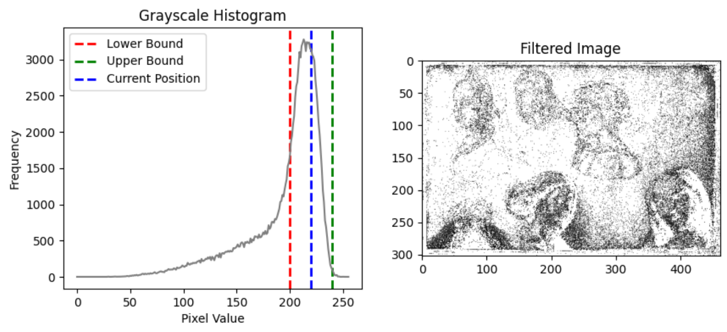


Figure 3.1: Range: 200-240

After trying different intervals, The effect is still not good. These features are mixed together, and it is not possible to filter some features separately by gray-scale filtering. It also requires manual setting for each image, which is a huge and time-consuming works to do manually for every image in the datasets.

Then I also tried to find the drawings by using OpenCV corner detection[13]. It can find part of the drawings, such as shadows that do not have strongly contrast with the surrounding pixels cannot be identified. After deleting the pixels of the drawings, I fill in the missing parts using the average pixel colors around them, as shown in the figure 3.2. However, the effect is still very poor. It eliminate some drawings and also leave other new features.

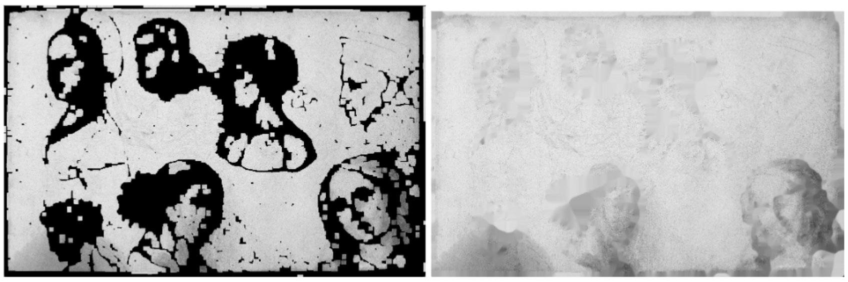


Figure 3.2: Filter by Canny corner detection

Furthermore, I utilized the content-aware fill feature in Photoshop to remove the drawings from the image, retaining only the water stains. This functionality is based on the PatchMatch[14] algorithm. A brief overview of its functionality involves searching for similar regions in the image and using those regions to replace the removed pixels. As illustrated, the algorithm consists of three steps: initialization, propagation, and search; as shown in the figure 3.3.

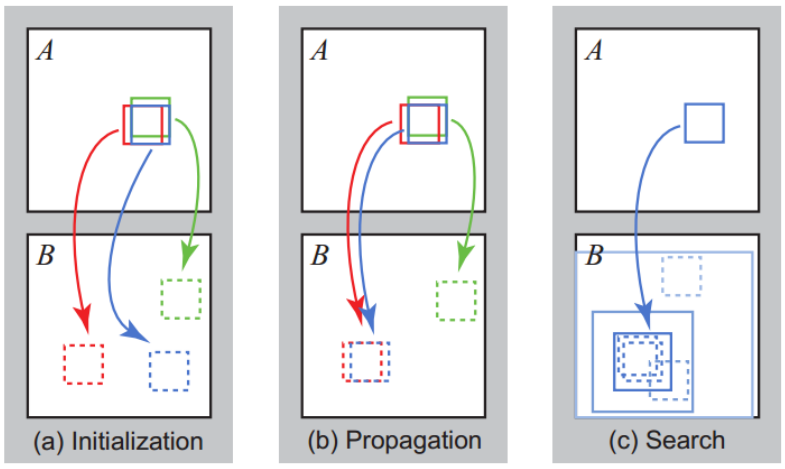


Figure 3.3: Patchmatch

In the initialization step, each patch is randomly assigned to another patch. Subsequently, iterations begin, with each iteration involving a global scan. Odd iterations proceed from top to bottom and left to right, while even iterations proceed in the opposite direction. The unit of scanning is a patch, and each patch undergoes two steps during scanning: propagation and search.

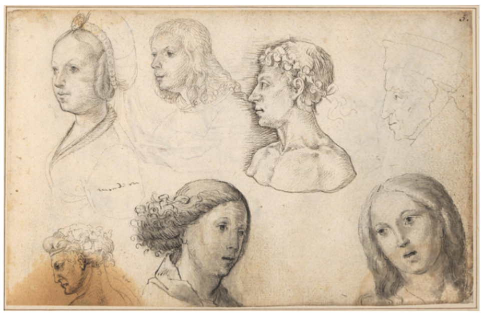
Propagation: Each patch attempts to borrow the match from its neighbors, choosing the neighbor′s match if it is better than its own. For example, during an odd iteration while scanning at position (x, y), the left neighbor $(x - 1, y)$ and the upper neighbor $(x, y - 1)$ of the current patch have already been scanned. The offsets corresponding to these neighbors are $f(x - 1, y)$ and $f(x, y - 1)$, respectively. These offsets represent the positions of their matching patches relative to themselves in the target image B. The current patch selects the best match among these three. The formula is as follows, where D represents the matching error between the patch in image A and the patch in image B.

$$f\left(x,y\right) = argmin_f\left[D\left(f\left(x,y\right)\right), D\left(f\left(x-1,y\right)\right), D\left(f\left(x,y-1\right)\right)\right]$$

Search: Although a matching patch has been found, it may not be the best result. Therefore, the algorithm needs to escape local optima and search for better results. Starting from the current matched patch $(x, y) + f(x, y)$, within a continuously decaying radius, it randomly matches several times until the condition is met. The formula is as follows,

$$v_i = f\left(x,y\right) + \omega\alpha^i R_i$$

Where ω is the maximum search radius, α is the decay factor between 0 and 1, and R is a pseudo-random number uniformly distributed in $[-1, 1] \times [-1, 1]$. Here, i denotes the number of random searches, and v represents the relative position of the patch in image B after the i-th random search. Here 3.4b is the result of Patchmatch.


(a) Original

(b) Result

Figure 3.4: Photoshop's result

Then, I used algorithms to calculate the similarity between processed images by comparing them. Average hash(Ahash) is common hashing algorithms used for image similarity comparison and retrieval[15], [16]. The Ahash algorithm scales the image to a fixed size (such as 8x8), then converts the pixel values to gray-scale values, and calculates the average of the pixel values. Next, the image is converted to a binary image (0 or 1), depending on whether the pixel value is greater than the average, and a 64-bit hash is generated[15]. Then, 14 consecutive images are extracted From the sketchbook to a test set (from page E3 to E13v), and the similarity between each image and other images is calculated to form a similarity matrix. This is the result of 14 images. Table 3.1 is the result of the original image used to find the verso image, and table 3.2 is the mirror of the target image used to find the adjacent pages. By looking at the matrix, for the first matrix, there are only 6 correct matches: 7, 8, 9, 10, 11, 12, 14, giving an accuracy of 0.5. The second matrix only gets 9 and 12 correct, which is 0.143. This accuracy is not enough, and it can be seen that Hash can not deal with this data set well.

Table 3.1: Origin

1	0.60546875	0.53515625	0.606445313	0.517578125	0.586914063	0.622070313	0.578125	0.711914063	0.548828125	0.662109375	0.541015625	0.569335938	0.522460938
0.60546875	1	0.5078125	0.631835938	0.466796875	0.520507813	0.618164063	0.5390625	0.676757813	0.478515625	0.65234375	0.490234375	0.604492188	0.526367188
0.53515625	0.5078125	1	0.528320313	0.62890625	0.604492188	0.569335938	0.65625	0.534179688	0.65234375	0.5	0.673828125	0.504882813	0.651367188
0.606445313	0.631835938	0.528320313	1	0.493164063	0.599609375	0.640625	0.592773438	0.69140625	0.538085938	0.688476563	0.571289063	0.615234375	0.58203125
0.517578125	0.466796875	0.62890625	0.493164063	1	0.604492188	0.512695313	0.619140625	0.500976563	0.642578125	0.455078125	0.646484375	0.481445313	0.616210938
0.586914063	0.520507813	0.604492188	0.599609375	0.604492188	1	0.623046875	0.670898438	0.62109375	0.700195313	0.559570313	0.706054688	0.501953125	0.6484375
0.622070313	0.618164063	0.569335938	0.640625	0.512695313	0.623046875	1	0.608398438	0.751953125	0.575195313	0.694335938	0.586914063	0.6171875	0.568359375
0.578125	0.5390625	0.65625	0.592773438	0.619140625	0.670898438	0.608398438	1	0.616210938	0.734375	0.56640625	0.73046875	0.536132813	0.653320313
0.711914063	0.676757813	0.534179688	0.69140625	0.500976563	0.62109375	0.751953125	0.616210938	1	0.573242188	0.772460938	0.581054688	0.650390625	0.5703125
0.548828125	0.478515625	0.65234375	0.538085938	0.642578125	0.700195313	0.575195313	0.734375	0.573242188	1	0.486328125	0.75390625	0.493164063	0.682617188
0.662109375	0.65234375	0.5	0.688476563	0.455078125	0.559570313	0.694335938	0.56640625	0.772460938	0.486328125	1	0.537109375	0.624023438	0.545898438
0.541015625	0.490234375	0.673828125	0.571289063	0.646484375	0.706054688	0.586914063	0.73046875	0.581054688	0.75390625	0.537109375	1	0.510742188	0.702148438
0.569335938	0.604492188	0.504882813	0.615234375	0.481445313	0.501953125	0.6171875	0.536132813	0.650390625	0.493164063	0.624023438	0.510742188	1	0.5234375
0.522460938	0.526367188	0.651367188	0.58203125	0.616210938	0.6484375	0.568359375	0.653320313	0.5703125	0.682617188	0.545898438	0.702148438	0.5234375	1

Table 3.2: Mirror

1	0.533203125	0.599609375	0.569335938	0.5625	0.670898438	0.588867188	0.650390625	0.635742188	0.65625	0.55859375	0.626953125	0.528320313	0.573242188
0.533203125	1	0.64453125	0.504882813	0.6171875	0.598632813	0.549804688	0.63671875	0.551757813	0.673828125	0.47265625	0.66796875	0.497070313	0.614257813
0.599609375	0.64453125	1	0.659179688	0.484375	0.551757813	0.616210938	0.56640625	0.661132813	0.494140625	0.66015625	0.533203125	0.645507813	0.569335938
0.569335938	0.504882813	0.659179688	1	0.635742188	0.634765625	0.60546875	0.659179688	0.58984375	0.676757813	0.541992188	0.678710938	0.51953125	0.6328125
0.5625	0.6171875	0.484375	0.635742188	1	0.518554688	0.600585938	0.52734375	0.633789063	0.4609375	0.65625	0.484375	0.575195313	0.516601563
0.670898438	0.598632813	0.551757813	0.634765625	0.518554688	1	0.693359375	0.579101563	0.705078125	0.551757813	0.708007813	0.594726563	0.591796875	0.56640625
0.588867188	0.549804688	0.616210938	0.60546875	0.600585938	0.693359375	1	0.692382813	0.65625	0.717773438	0.581054688	0.702148438	0.533203125	0.638671875
0.650390625	0.63671875	0.56640625	0.659179688	0.52734375	0.579101563	0.692382813	1	0.737304688	0.5703125	0.73828125	0.611328125	0.590820313	0.594726563
0.635742188	0.551757813	0.661132813	0.58984375	0.633789063	0.705078125	0.65625	0.737304688	1	0.760742188	0.569335938	0.721679688	0.560546875	0.64453125
0.65625	0.673828125	0.494140625	0.676757813	0.4609375	0.551757813	0.717773438	0.5703125	0.760742188	1	0.77734375	0.537109375	0.645507813	0.530273438
0.55859375	0.47265625	0.66015625	0.541992188	0.65625	0.708007813	0.581054688	0.73828125	0.569335938	0.77734375	1	0.7578125	0.475585938	0.682617188
0.626953125	0.66796875	0.533203125	0.678710938	0.484375	0.594726563	0.702148438	0.611328125	0.721679688	0.537109375	0.7578125	1	0.637695313	0.584960938
0.528320313	0.497070313	0.645507813	0.51953125	0.575195313	0.591796875	0.533203125	0.590820313	0.560546875	0.645507813	0.475585938	0.637695313	1	0.583984375
0.573242188	0.614257813	0.569335938	0.6328125	0.516601563	0.56640625	0.638671875	0.594726563	0.64453125	0.530273438	0.682617188	0.584960938	0.583984375	1

4. Neural Network Approach

My research fields are in artificial intelligence and computer vision, two fields that have made tremendous progress in recent years. There are many related products that have been very popular recently, such as ChatGPT, an AI language model developed by OpenAI that is trained to understand and generate natural language text. For example, Apple's Face ID and Google's face recognition technology are computer vision based products. Therefore, I want to use the professional knowledge I have learned to try to solve the problem of sketchbook re-ordering with the help of AI and computer vision.

For this project, I will solve this problem using computer vision and neural networks. Computer vision is used to obtain the features of the image, and then this data is fed into the neural network and the model is trained. Finally, we rely on the trained model to find the relationship between the pictures in the sketchbook. Eventually, we can rely on this model to solve other similar problems, which will greatly reduce the time spent by the researchers involved.

This project mainly consists of four steps: firstly, training-set creation; secondly, constructing a Yolov8 and training the model; thirdly, extracting information about water stains from the images using the trained model; finally, utilizing this information to restore the original order of the sketchbook.

4.1 Training-set creation

As mentioned earlier, re-ordering relies on the positioning, size, and other characteristics of water stains. Therefore, the Yolov8 is responsible for identifying the features of water stains. When creating the training dataset, it is necessary to specify the required information in the images, such as the category, size, and position of stains, which requires manual labeling. In this regard, the open-source tool – Roboflow[17] is adopted for creating the training dataset. The figure 4.1 displays an example.

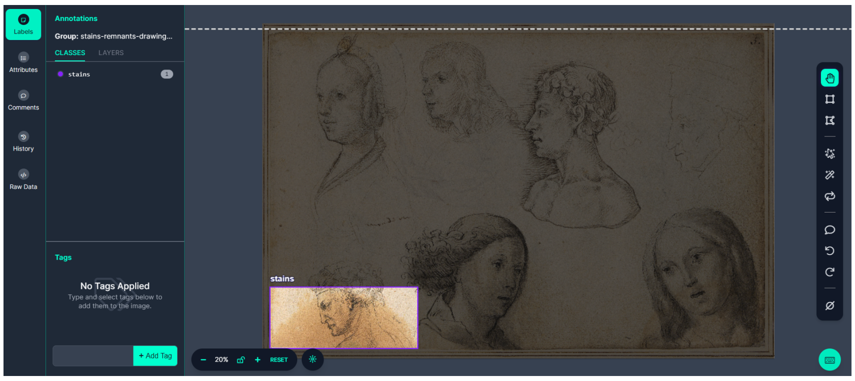


Figure 4.1: Roboflow

By manually annotating the features in the images and exporting the data, the dataset is structured into three arrays: 'Images,' 'Categories,' and 'Annotations.' Each element in the 'Images' array contains the index, size, and file path of the image. Each element in the 'Categories' array contains the index and name of the label. The 'Annotations' array contains all feature information, with each element corresponding to a feature. This includes the feature's index, the index of the image it belongs to, the index of the label, and the bounding box's position and size information.

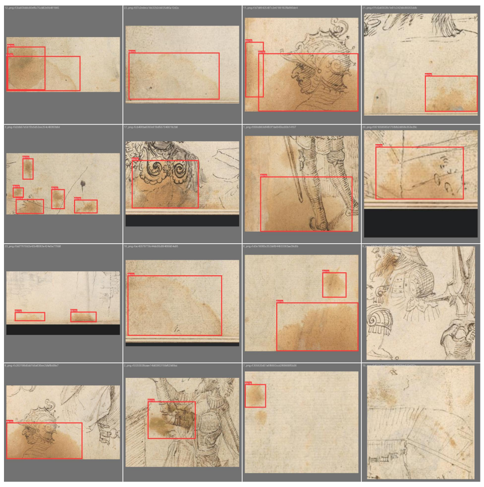
4.1.1 Training set augmentation

A total of 55 images containing stains were filtered from the dataset. However, the dataset is too small, and there needs to be expended. In this regard, the most commonly used method for data augmentation is to apply transformations to the original images, such as rotate 90 degrees clockwise, horizontal flip. Relevant studies[18] have shown significant improvements in various versions of YOLO using this approach. Through this table 4.1, the effect of data augmentation can be intuitively observed. Here, OD represents the original dataset, and AD represents the augmented dataset.

Table 4.1: Data augmentation's impacts on different versions of Yolo, as measured by precision, recall, and F1-score [18].

Models	Precision		Recall		F1-sore	
	OD	AD	OD	AD	OD	AD
YOLOv5	0.980	0.955	0.853	0.913	0.912	0.934
YOLOv6	0.967	0.948	0.530	0.920	0.685	0.934
YOLOv7	0.769	0.972	0.827	0.963	0.797	0.967
YOLOv8	0.685	0.994	0.760	0.965	0.721	0.979

First, the original dataset was split into training and validation sets in a ratio of 9:1. To avoid errors due to different images in the test set when evaluating the model's accuracy, the test set comprised a fixed set of images. These images were obtained by cropping certain images from the Errera's, and all of them contain water stains. The bounding box of the water stains in the images were manually annotated. In total, there are 23 images, as illustrated in the figure 4.2.


(a)

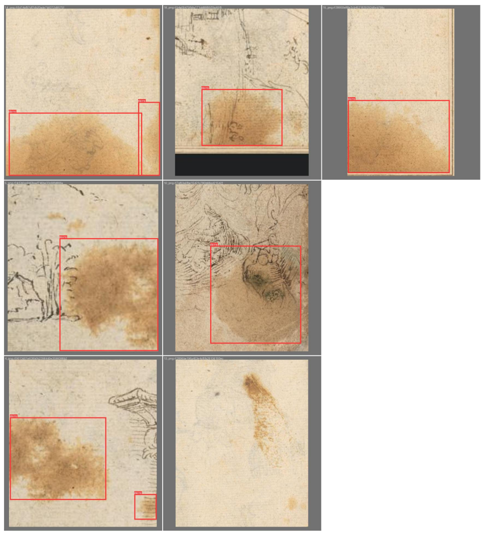

(b)

Figure 4.2: Test set

Data augmentation was then applied to the training set, including image rotation and horizontal flipping. Consequently, the final dataset size was $trianing \times (1 + n) + validation$, where n represents the number of operations. The validation set was not augmented since it is not involved in the training process; it is only used to validate the model and adjust the learning rate after each epoch.

4.1.2 DALL-E generate

DALL-E Mini [19] is an image generation tool based on OpenAI's model. It generates images based on prompts. Using features from the Errera sketchbook as prompts, such as "sketching," "yellowing," and "water stains," it generated a total of 50 images, as shown in the figure 4.3.

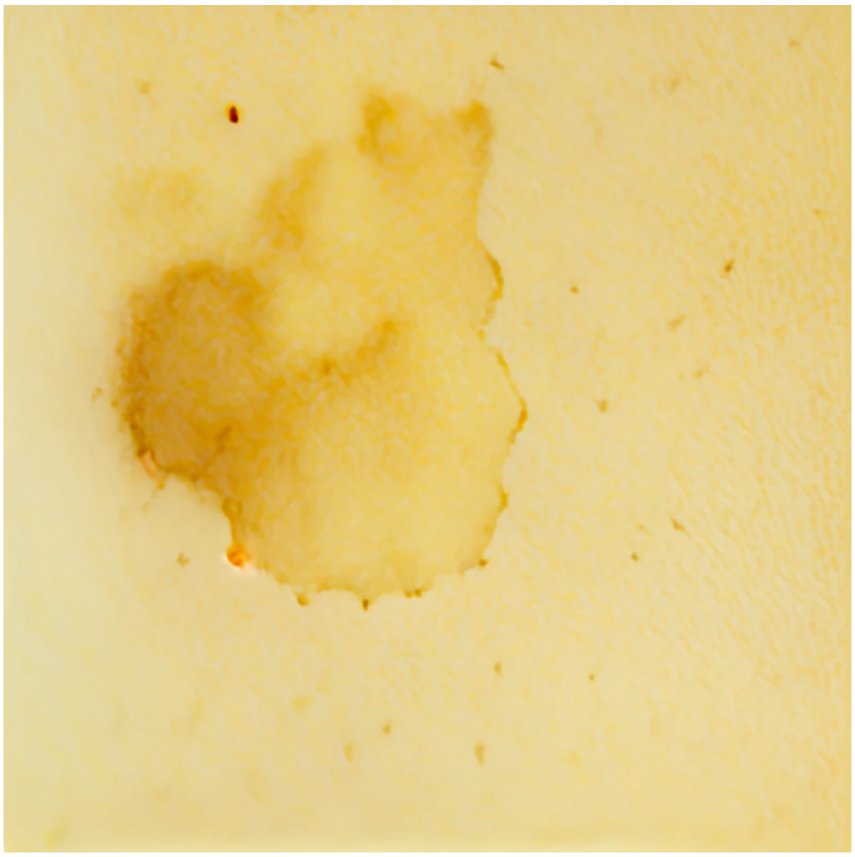
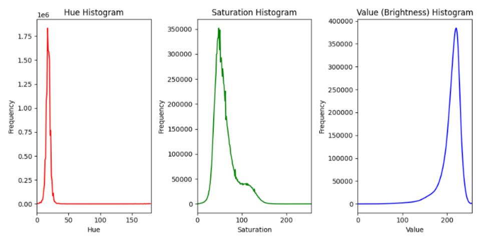
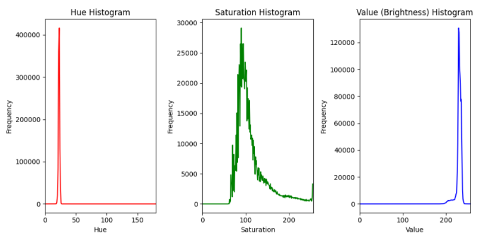


Figure 4.3: DALL-E's generated

However, the color style of these images differs from the Errera sketchbook. By comparing their HSV color spaces, adjustments can be made to align them more closely. For example, by observing its HSV color space and comparing it with the Errera sketchbook's HSV color space 4.4,



(a) Errera's HSV histograms



(b) DELL-E's HSV histograms

Figure 4.4: Histograms of both in HSV color space

we can notice that the AI-generated image tends to be shifted to the right in all three channels. To correct this, I applied a leftward offset to each channel. The resulting images, with the adjusted color style, are shown in the figure 4.5.

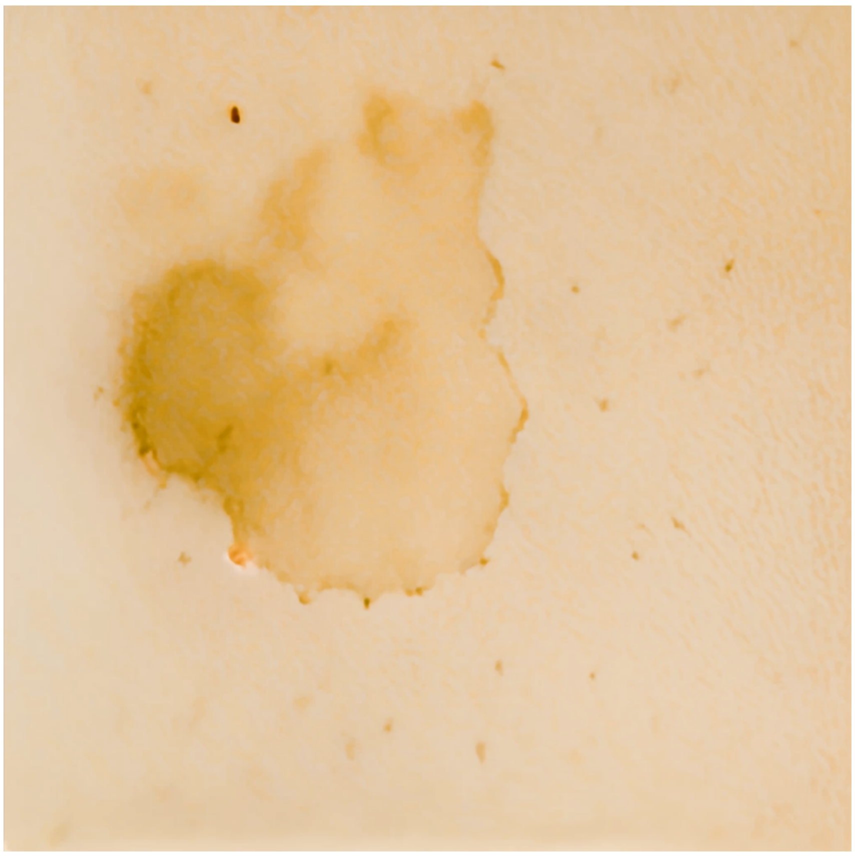


Figure 4.5: After changed

4.2 Simulation of ink diffusion

Essentially, this is simulating the diffusion process of water molecules on paper as they move from high concentration to low concentration along the fibers [20], [21]. There are two main parts for the simulation: modeling paper texture and modeling water diffusion.

In the modeling paper texture, the paper is consider as a 2D array, each element is a node, which has a position, corresponding neighbors, material type(space, fiber, fiber node), and current water capacity [21]. The pseudo code is as follows.

```
class Node {
  Position: (x, y)
  Up: Node(x, y-1)
  Down: Node(x, y+1)
  Left: Node(x-1, y)
  Right: Node(x+1, y)
  Texture: 0 - Empty, 1 - Fiber, 2 - Fiber Node
  Capacity: C
}
```

Then, randomly generate line segments of varying lengths and directions. The endpoints are fiber nodes, and the middle sections are fibers[20]. These are then mapped onto the 2D array.

This is a water diffusion simulation based on cellular automata, where each iteration updates the state of each node. Therefore, it requires information about each node and its surrounding nodes, including material type (T), bottom height (Bheight), current storage capacity (C), diffusion rate (a), and minimum diffusion value (minDiff). The original approach [21] considered both outward diffusion from each node and inward absorption from surrounding nodes. The pseudo code is as follows.

```
While(!Stop)
  For node n in Nodes:
    amount = 0
  If n.T = 0 or n.C < minDiff:
    Continue;
  For neighbor k in n.neighbors(Up, Down, Left, Right):
    k_height = k.Bheight + k.C
    n_height = n.Bheight + n.C
    larger_capacity = max(k.Bheight, n.Bheight)
    k2o = max(0, 0.25 * DEFFUSION_RATE *
          min(k_height - n_keight, k_height - larger_capacity))
    o2k = max(0, 0.25 * DEFFUSION_RATE *
          min(n_height - k_height, n_height - larger_capaticy))
    amount += (k2o - o2k)

  n.C += amount
```

4.2.1 Stains initialization

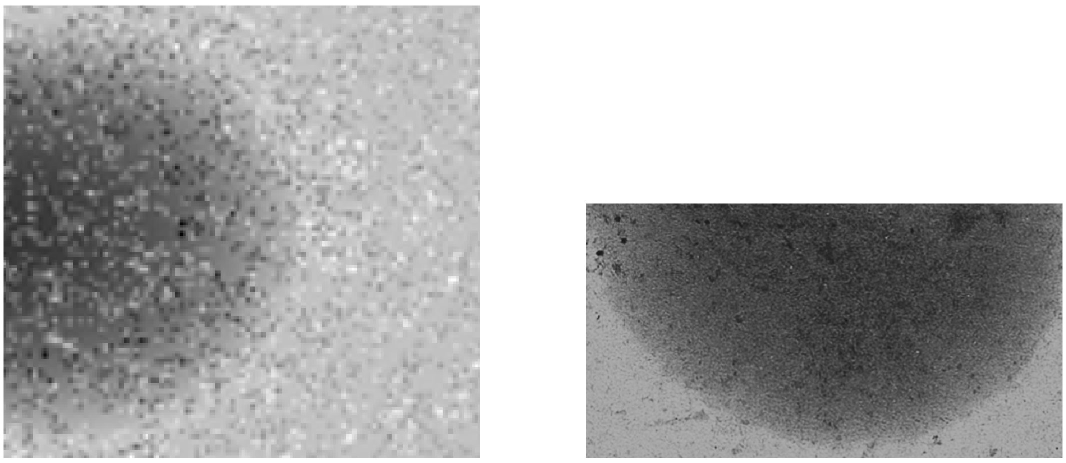
This section requires initializing the shape, size, intensity, and position of the water stain. The initial shapes include circles and ellipses, where their positions, radii, and intensities are all random values. Additionally, to ensure the irregularity of the shapes of circles and ellipses, an offset Irregularity (IR) is added to any point on the boundary. Finally, any pixel point P in a circle satisfies the following equation:

$$\sqrt{(Center\_x - x)^2 + (Center\_y - y)^2} \leq r \cdot (1 + \text{random}(-\text{IR}, \text{IR}))$$

Another type of image is an irregular radial pattern. Starting from a central point, rays are emitted in random directions and with random lengths. Then, these points are connected to form a closed polygon. Finally, the Catmull-Rom Splines[22] are used to process the boundary into a smooth curve.

4.2.2 Simulation result

The next step is to perform the diffusion simulation based on the initial shapes. The following images 4.6 show the simulation results after multiple iterations.



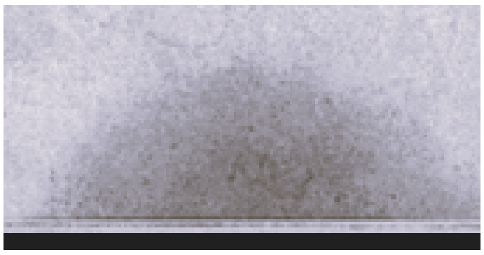
(a) Simulation result **(b)** Errera's example

Figure 4.6

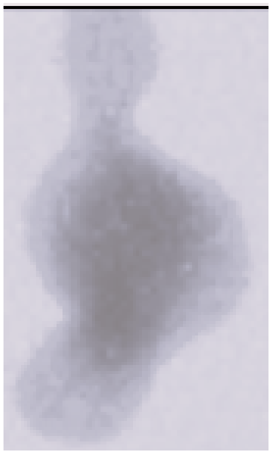
We compared the generated images with water stains in the sketchbook. The generated images include the characteristics of real water stains. For example, intensity gradients, stronger coloration at fiber nodes, weaker coloration along fibers, and water molecules spreading along the fibers. However, this simulation also has some shortcomings, such as the inability to simulate different speeds in different directions, high resolution, and the accumulation effect of water molecules at boundary. The table below shows the compared in characteristics.

Characteristics	Errera's	Simulation's
Diffusion speed in different direction	T	F
Intensity gradients	T	T
Stronger coloration at fiber nodes	T	T
Weaker coloration at fiber	T	T
molecules spreading along the fibers	T	T
High resolution	T	F
Irregular shape	T	T

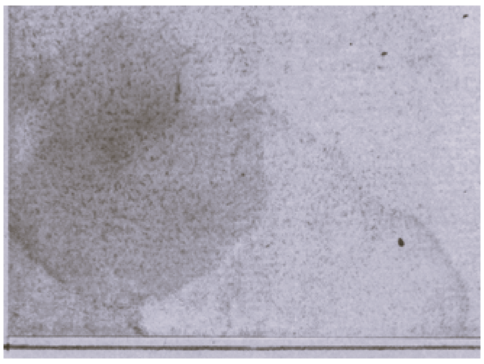
(a) Simulation result



(b) Errera's example



(c) Simulation result



(d) Errera's example

(e) Simulation result

(f) Errera's example

Figure 4.7: Here are some generated examples compared to the original image.

Finally, by simulation, the number of images in the original data set was increased to 708 for training and 73 for validation. The test set uses a fixed 10 pictures containing water stains from Errera's sketchbook. After data augmentation, the final dataset size is 2197.

4.3 Re-ordering

Daantje Meuwissen, who is the expert of arts historian in Northern Europe, pointed out that the corresponding Recto and Verso relationships in the sketchbook are correct; they indeed appear on the same sheet of paper, front and back, respectively. Therefore, there is no need to reorder them. Instead, these pairs should be stored as paired data, with each data type being [recto image, verso image]. By traversing the entire sketchbook and pairing the images based on their names, they are stored in a list named MatchedImages.

Subsequently, Yolov8 is used to predict the bounding boxes of water stains in the images within MatchedImages, outputting a 2D array - [[recto-bboxes], [verso-bboxes]], which is stored in a list named BBoxes. For each bounding box within an element, the Intersection over Union (IoU) with the bounding boxes needs to be calculated. This metric measures the overlap between two water stains: the higher the IoU, the greater the overlap area. The best match is selected based on the highest average IoU, marking the element for the next search iteration and removing the current element. If the current element cannot find a match, it is skipped and the process moves to the next element. The pseudo-code for this process is as follows:

```
def match(MatchedImage, page2match):
  if page2match does not contain any bbox:
    return false
  average_IOU = 0
  for page in MatchedImage:
        current_average_IOU = 0
        if page is same to page2match:
          skip to next iteration
        # search for recto
    for recto_bbox in page2match's recto_bboxes:
          current_recto_IOU = 0
          for current_recto_bbox in page's recto_bboxes:
            IOU = Calculate IOU between recto_bbox
```

```
                and current_recto_bbox
                if IOU > current_recto_IOU:
                  current_recto_IOU = IOU
                Number_matched += 1
                total_IOU += current_recto_IOU

        # search for verso is the same to recto

        current_average = total_IOU/Number_matched
        if current_average > average_IOU:
          average_IOU = current_average
          mark current page as best_match
        return best_match

def re_ordering():
  while the page2match is not last page in MatchImage:
    page2match = match(MatchImage, page2match)
        if page2match does not contain any bbox:
          set next page as page2match
        else:
          save page2match to result
          remove the page2match from MatchImage
```

5. Experiment

The first of all, we have to figure out which type of image is appropriate for the project. I build two train sets, color and gray. Then, to test the accuracy of the model trained by different datasets. This is used to evaluate the impact of gray and color images on accuracy. Each model is trained for 40 epochs. Each epoch, all images in training set will be learned by the model once, so epochs means how many times the model is trained. The number of epochs was set to 40 solely for the purpose of testing whether color or gray images yield better results, rather than for training a final model. Below 5.1 are the accuracy obtained for different datasets.

Dataset(number of images)	Errera's color like	Grey
Errera (142)	0.37	0.39
Errera and DALLE(207)	0.54	0.43

Table 5.1: Accuracy for different data set

From the table 5.1 above, it can be observed that the accuracy significantly increased after expansion, and the accuracy for color images was higher than that for gray images. To summarize, color images are better than gray images for training a model in this project.

Next, a detailed analysis of the expanded dataset is provided, as illustrated in the figures 5.1. These charts represent the bounding box loss and classification loss for both the training set and the validation set. Overall, there are some fluctuations, and the trends are decreasing but have not stabilized. This indicates that the number of epochs used in the model training is still insufficient.

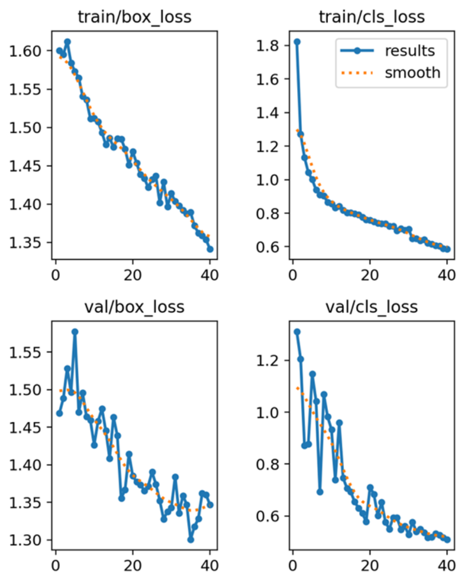


Figure 5.1: 40 Epochs

I set the maximum number of epochs to 1000 and employed an early stopping strategy with a patience parameter of 100. This means that the training process can run for up to 1000 epochs, but it will terminate early if there is no improvement in model accuracy for 100 consecutive epochs. And there is a new result, as shown in the figure 5.2

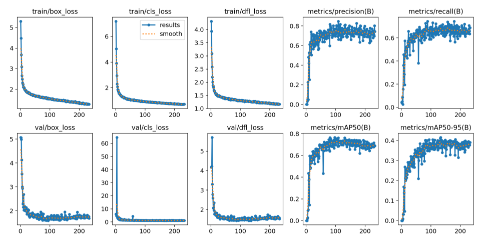
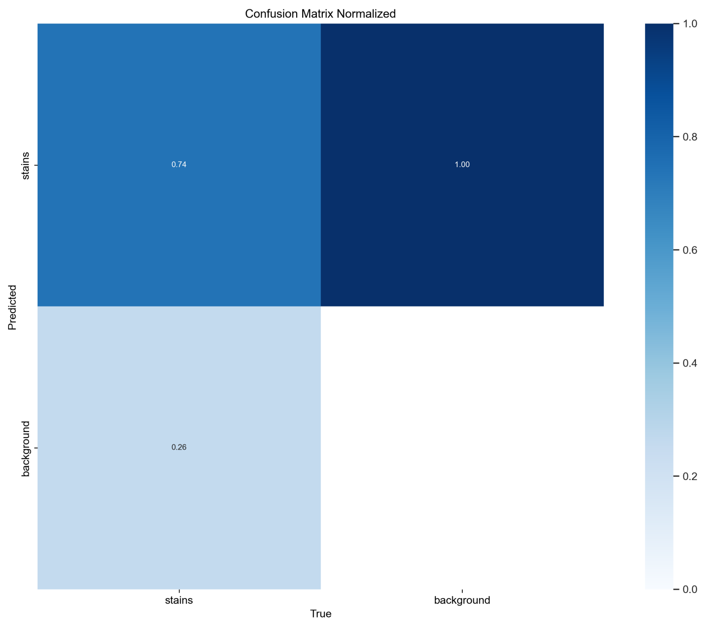
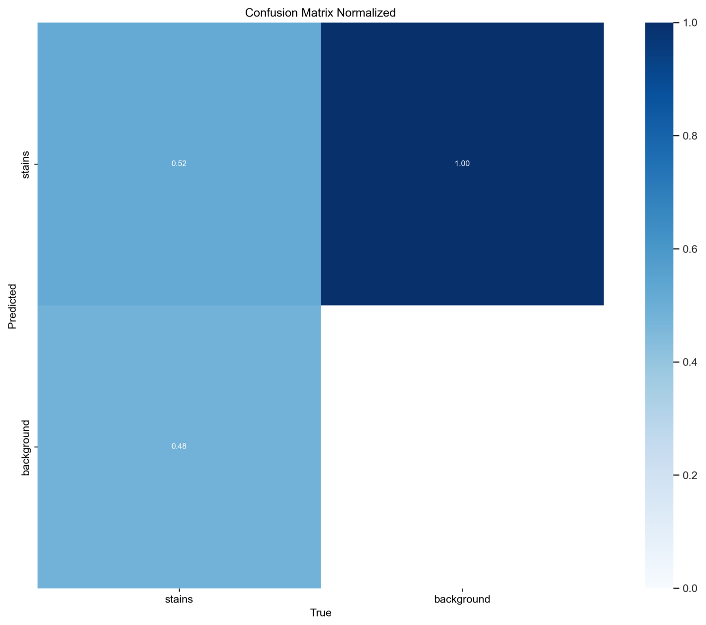


Figure 5.2: Accuracy of early stop at 234th epoch

Although these loss function curves are more stabilized, the accuracy on the test set is only 0.52. As shown in the figure 5.3.



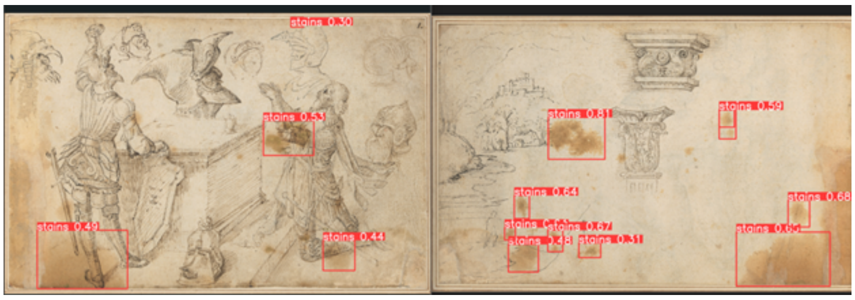
(a)



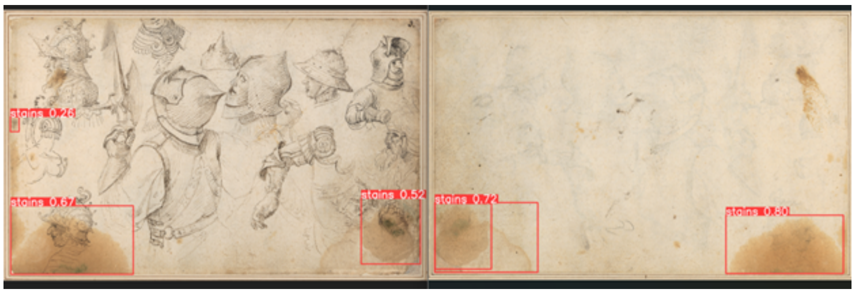
(b)

Figure 5.3: Accuracy of 234 epochs on the valid set and test set

According to this model, some water stains in the sketchbook will be detected and record. By Re-ordering, each image was matched to find its adjacent page. Here is an instance, E1 and E1V were used as targets for the search, returning results are E3 and E3V. As shown in the figure 5.4, these two images appear to be adjacent based on the water stains, indicating a good result.



(a) E1 and E1V



(b) E3 and E3V

Figure 5.4: Matched result by Re-ordering

However, when encountering images with only a few detected water stains, insufficient information is available for matching, leading to failure. As shown in the figure 5.5, the model was able to find only three water stains in the images, one of which was a false detection. To further improve the model's accuracy, this project will implement enhancements in both the dataset and the Yolov8 architecture.

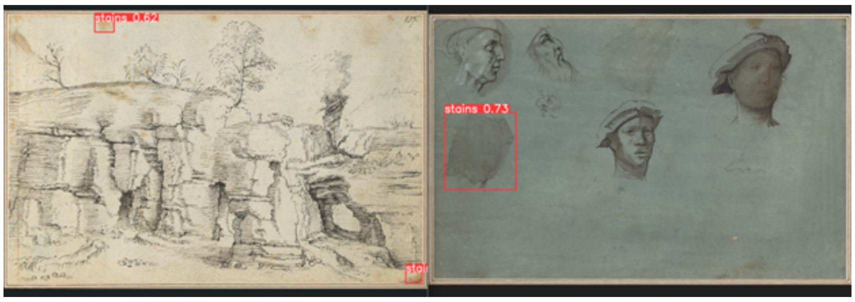
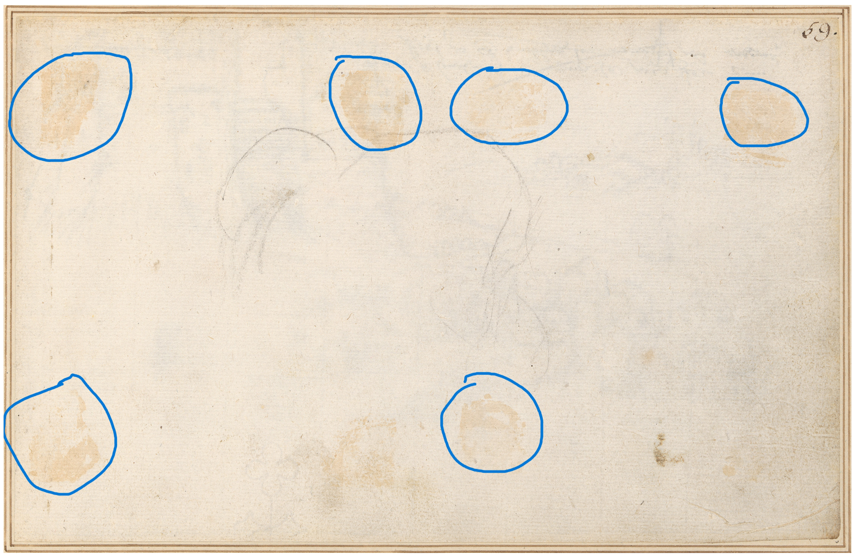


Figure 5.5: Insufficient water stains detected

5.1 Dataset improvement

The results so far were evaluated with the art history expert, she pointed out that there were issues with the bounding boxes in the current dataset. Some water stains were missed, while others were incorrectly identified as water stains. With her assistance, I corrected these errors. For instance 5.6, the marks in the image are not water stains.


(a) Caused by tape

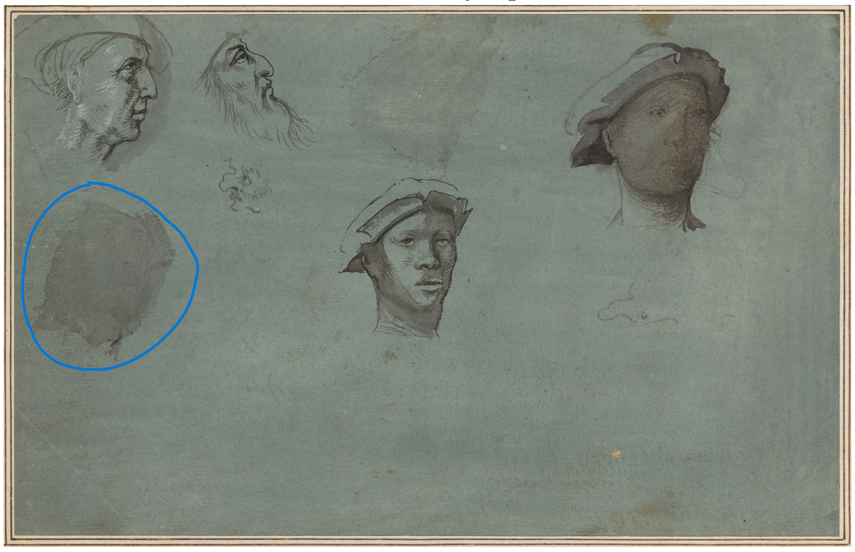
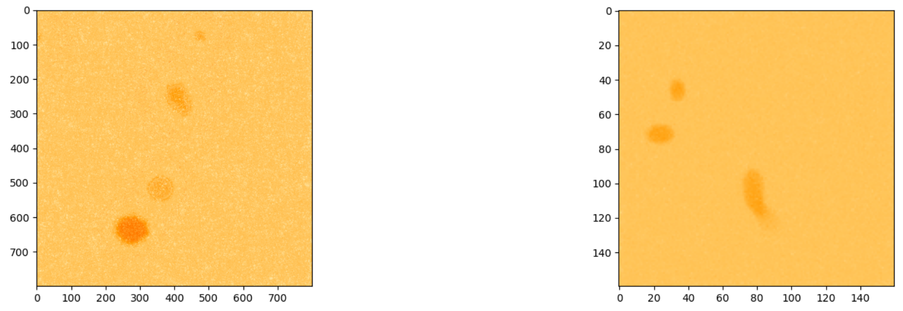

(b) Artist work

Figure 5.6

On the other hand, the art history expert also pointed out several deficiencies in the simulated images. First, the fiber density in the images is too high and should be reduced, as illustrated in the figure 5.7.



(a) 50000 **(b)** 100000

Figure 5.7: Different density in resolution of 800*800

Second, unlike in Errera's images, the simulated water stains appear in the center of the image rather than at the edges. Lastly, the overall color of the images is too orange; it would be better to overlay the simulated water stains directly onto the Errera images. Based on the art history expert's feedback, I updated the simulation algorithm. I reduced the fiber density and saved the simulated water stains as a 2D array to record the concentration at each position. Finally, I determined the color intensity based on the concentrations in the array and drew the stains along the edges of the Errera images. Ultimately, 149 new images were generated, and after removing those with poor results, a total of 127 images remained, There is an example 5.8 for new simulation.

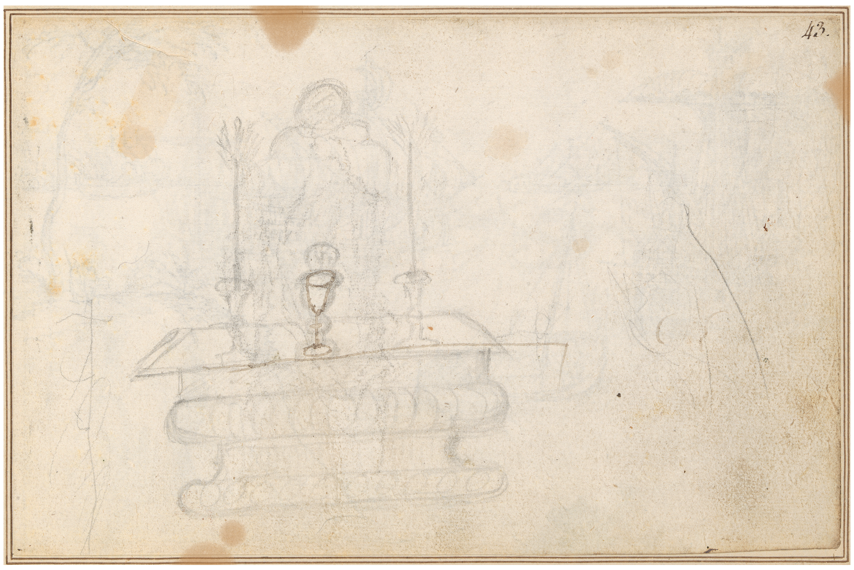


Figure 5.8: New simulation

Compared to the original results, the improved simulation based on the art history expert's suggestions yielded better outcomes. The colors are more closely aligned with the effects observed in the sketchbook, and the positions of the water stains now appear at the edges of the images, more accurately simulating the original water stain effects. Furthermore, the density of the fibers in the simulation has decreased, allowing for a clearer observation of the diffusion of water stains between the fibers.

5.1.1 Model performance on different data sets

After correcting the bounding boxes in the dataset and creating new simulated water stains, I constructed several new datasets and trained new models, testing their performance under different parameters. These parameters included whether to use a pre-trained model. The pre-trained model is sourced from Ultralytics[9]. It has been trained on the COCO dataset[23] to accelerate convergence in other model training tasks. Within the limit of 1000 epochs, early stopping was employed (the model records the accuracy of each training session and stops if there is no improvement in the last 100 epochs). Finally, I tested their performance on the validation set and the Errera sketchbook. The results are shown in the table below 5.2.

Dataset	Number of images	Pre-training	Early stop	Accuracy on valid	Accuracy on test
E	142	T	218	0.87	0.7
E	142	F	256	0.67	0.49
E B DL O OS NS	1983	T	215	0.87	0.83
E B DL O OS NS	1983	F	243	0.91	0.7
E B DL O OS NS(+150)	2126	T	230	0.74	0.39
E B DL O OS NS(+150)	2126	F	297	0.77	0.32
E NS(+150)	464	T	138	0.86	0.65
E NS(+150)	464	F	233	0.74	0.52

Table 5.2: Accuracy for different data set. E: Bounding boxes in Errera Corrected by the art history expert; NS: new simulation images; NS(+150): Additional 150 simulated images; B: Berlin Sketchbook; DL: DALLE mini; OS: Old simulation.

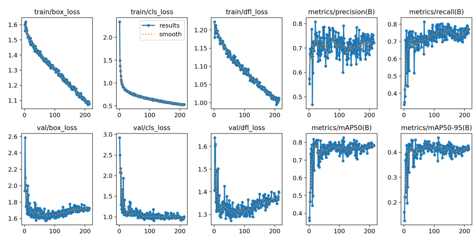
Abbreviation	Dataset's name	Source	Notes
E	Errera	Errera sketchbook	
B	Berlin	Berlin sketchbook	
DL	DALLE	Generated by DALLE-mini	
OS	Old simulation of water stains	Simulation of water diffusion	Independent simulation images, examples in figure 5.7
NS	New simulation of water stains	Simulation of water diffusion	The effect of projecting onto Errera's images, examples in figure 5.8
NS(150)	New simulation with extra images	Simulaiton of water diffusion	Extra images added to NS
O	Other	Rijksmuseum	Image contained similar water stains which find in Rijksmuseum

Table 5.3: Details for different datasets

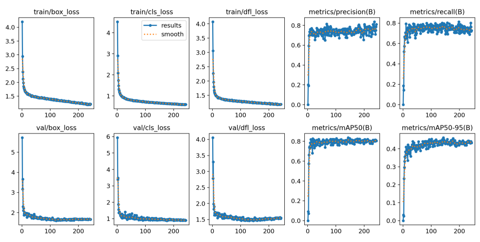
By observing the results, it is evident that using pre-training improves convergence speed and accuracy on both the validation set and Errera sketchbook for the first and fourth datasets compared to not using pre-training. On the other hand, for the second and third datasets, although pre-training accelerates convergence and improves validation set accuracy, it reduces the model's generalization ability, leading to lower accuracy on the Errera sketchbook.

Furthermore, when examining the performance of these datasets on the Errera sketchbook, it is clear that adding images from different sources can increase accuracy, indicating an improvement in the model's generalization ability. However, continuously increasing the number of images does not always improve accuracy; in fact, it can decrease it, as seen from the results of the third and fourth datasets.

Overall, among these datasets, only the second dataset performs exceptionally well, achieving an accuracy of 0.83 on the Errera test set. Here is a detailed analysis of the second dataset, as illustrated 5.9



(a) Pre-trained



(b) No pre-trained

Figure 5.9: Details included: box loss, class(cls) loss, distribution focal loss(dfl, help box loss to find better the bounding box), precision, recall, mean average precision of confidence under 50(mAP50), mean average precision of confidence of confidence between 50 and 95(mAp50-95)

When not using pre-trained models, the various metrics of the model eventually stabilized. However, when using pre-trained models, although the accuracy on the Errera sketchbook was better than without pre-training, the loss function did not converge, and the metrics exhibited significant fluctuations.

5.2 Yolov8 structure optimization

To integrate CBAM into the Yolov8 architecture while preserving its foundational structure, I insert the CBAM module after the second C2f block within the backbone. Subsequently, I decouple this C2f block from Concat layer in head module and connect it with CBAM, replacing the original C2f connection. The comparative results of this architectural adjustment are presented in the table 5.4.

Model	Pre-training	Early stop	Accuracy on valid	Accuracy on Errera
Yolov8	T	215	0.87	0.83
Yolov8	F	243	0.91	0.7
Yolov8+CBAM	T	202	0.92	0.9
Yolov8+CBAM	F	244	0.93	0.89

Table 5.4: Results of the two models on the dataset "E B DL O OS NS".

From this table, it can be found that combining CBAM in Yolov8 can improve the accuracy, whether it is the validation set or the test set.

5.3 Re-ordering result

This result is obtained based on the predicted bounding box information, utilizing a re-ordering algorithm. Detailed results can be found in the appendix 7.1. Although the reordering outcome for some images is correct, there are numerous cases of incorrect matches. These errors can primarily be attributed to the following factors:

Firstly, inaccuracies in the model's predictions. For example, after processing images E5-E5v, the algorithm incorrectly identifies E9-E9v as the next match instead of E7-E7v. This is primarily because the water stain in the bottom right corner of E7-E7v was not detected, leading to a lower score compared to E9-E9v, as illustrated in the figure 5.10.

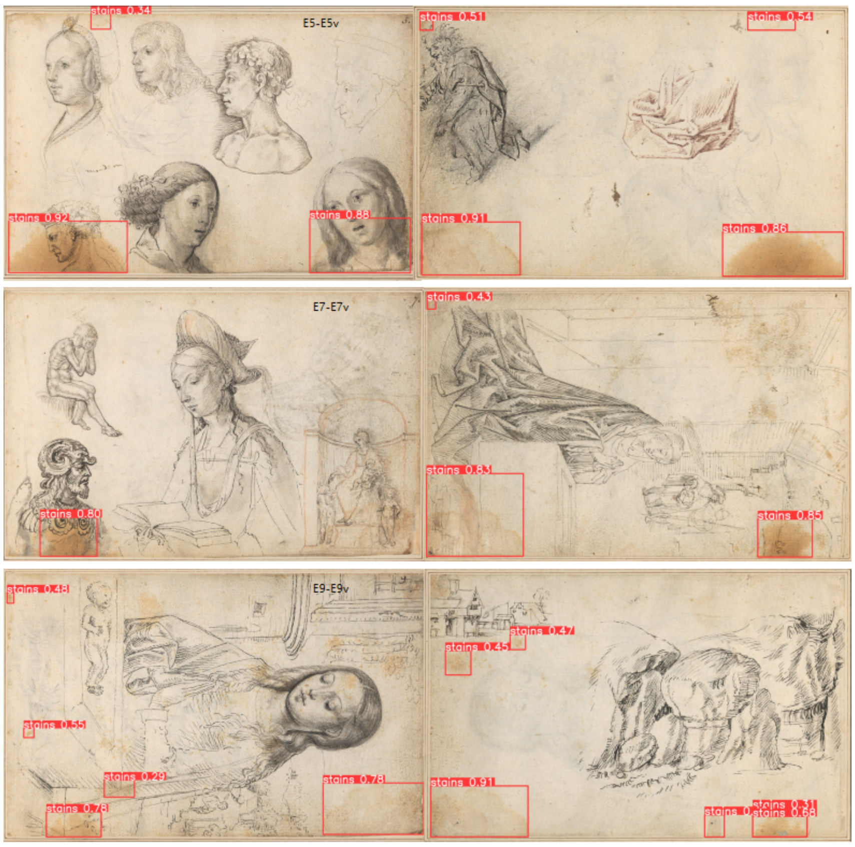


Figure 5.10: From top to bottom, the images are E5-E5v, E7-E7v, and E9-E9v. In E7, there is a water stain in the bottom right corner that is not detected.

Secondly, an excessive number of detected water stains in a single image can result in false matches. For instance, in the case of E163-E163v, numerous targets are detected, which lead to E11-E11v consistently matching with results from E163-E163v, yielding a high score and erroneously pairing E163-E163v with E11-E11v, shown in the figure 5.11.

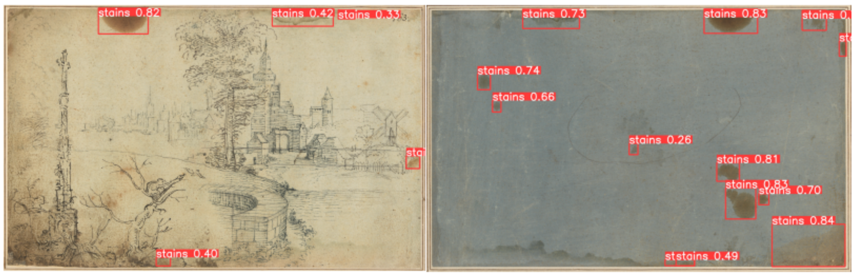


Figure 5.11: Page: E163-E163v, lots of water stains are detected.

On the other hand, some results produced by the algorithm may be correct. As shown in the figure 5.12, the page following E139-E139v in the sketchbook is E141-E141v. However, the algorithm suggests that the next page should be E147-E147v. By comparing the water stains and bounding boxes, we find that the algorithm's result is closer to the correct sequence than the original one.

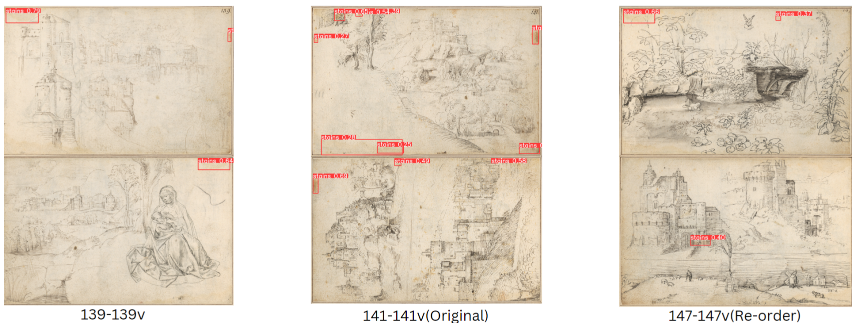


Figure 5.12: E139-E139v: the next page in the sketchbook and the next page given by algorithm.

6. Conclusion

This project aims to r-ordering the sequence of the Errera sketchbook with the assistance of AI. Based on a review of relevant literature, the study identifies water stains as a evidence in the re-ordering process. Although an initial attempt is made using basic image processing techniques to re-ordering the sequence, this approach proves inadequate for distinguishing features within the images and failed to achieve a successful reconstruction.

Subsequently, Yolov8 is employed to detect and predict water stains in the images. To enhance the model's accuracy, various strategies are implemented to augment the training dataset, including data augmentation, image generation, and the incorporation of similar images from other sketchbooks and museums. Additionally, the Yolov8 architecture is optimized by integrating the Convolutional Block Attention Module (CBAM), leading to improved model accuracy. However, the re-ordering algorithm still exhibits some limitations and is only effective for correctly sequencing a subset of the images.

6.1 Discussion

The study successfully identified water stains as a viable feature for re-ordering the Errera Sketchbook. However, while water stains provided a clear criterion for image matching in many cases, their presence alone was not sufficient to re-order all images accurately. This indicates that while water stains are a valuable piece of evidence, other features or a combination of multiple features might be necessary for a more comprehensive and accurate reconstruction.

7. Appendix A

7.1 Useful images

Name	Is Useful	Stains	Remnants
E 0a-B1	F	F	F
E 0b-B1	F	F	F
E 0c-B1	F	F	F
E 0d-B1	F	F	F
E 33v-B6	F	F	F
E 37v-B5	F	F	F
E 39v-B5	F	F	F
E 49v-B7	F	F	F
E 53v-B6	F	F	F
E cover1-B9	F	F	F
E cover2-B9	F	F	F
E cover3-B10	F	F	F
E cover4-B10	F	F	F
E spread1-B10	F	F	F
E spread2-B10	F	F	F
E spread3-B10	F	F	F
E 101-B1	T	T	T
E 101v-B1	T	T	T
E 102-B1	T	F	F
E 102v-B1	T	T	F
E 103-B1	T	F	F
E 103v-B1	T	F	F
E 105-B2	T	T	T
E 105v-B2	T	T	T
E 107-B2	T	T	F
E 107v-B2	T	T	T
E 109-B2	T	T	T
E 109v-B1	T	T	T
E 111-B1	T	T	T
E 111v-B1	T	T	F
E 113-B1	T	T	F
E 113v-B1	T	T	F
E 115-B1	T	T	T
E 115v-B1	T	T	T
E 117-B1	T	T	T
E 117v-B1	T	T	T
E 119-B1	T	T	F
E 119v-B2	T	T	F
E 11-B3	T	T	T
E 11v-B2	T	T	T
E 121-B2	T	T	T
E 121v-B2	T	T	T
E 123-B2	T	T	T

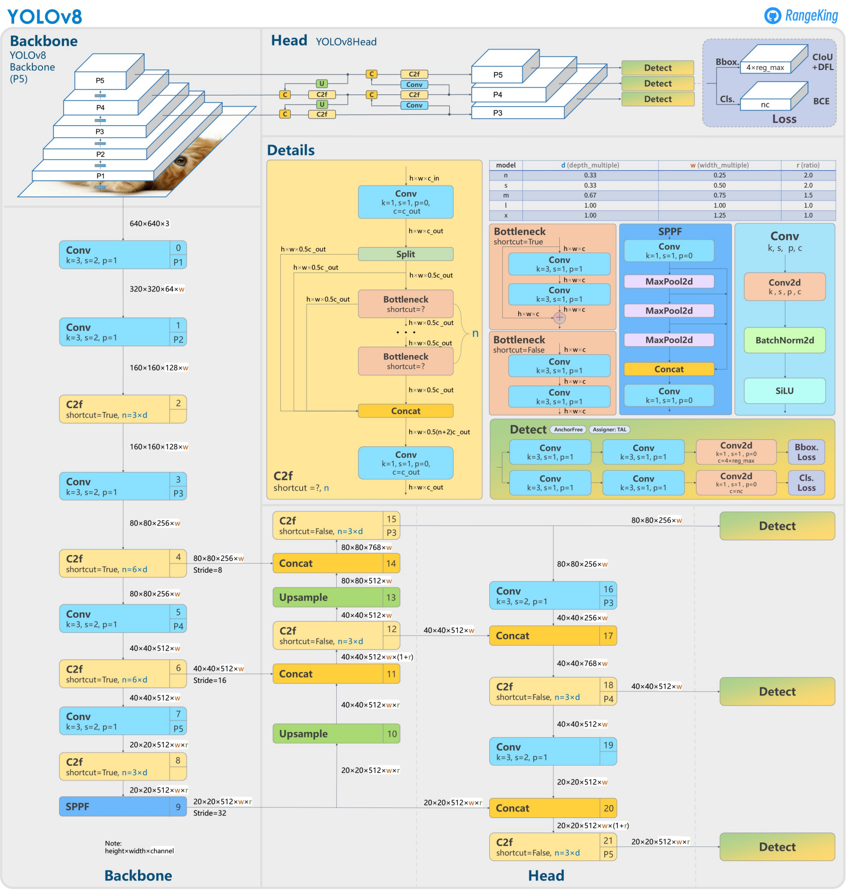
E 123v-B2	T	T	T
E 125-B2	T	T	T
E 125v-B2	T	T	T
E 127-B2	T	T	T
E 127v-B2	T	T	T
E 129-B2	T	T	T
E 129v-B2	T	T	T
E 131-B2	T	T	T
E 131v-B2	T	T	T
E 133-B2	T	T	T
E 133v-B3	T	T	T
E 135-B3	T	T	F
E 135v-B3	T	T	F
E 137-B3	T	F	F
E 137v-B3	T	F	F
E 139-B3	T	T	F
E 139v-B3	T	T	T
E 13-B3	T	T	T
E 13v-B3	T	T	T
E 141-B3	T	T	T
E 141v-B3	T	T	T
E 143-B3	T	T	T
E 143v-B3	T	T	T
E 145-B3	T	T	T
E 145v-B3	T	T	T
E 147-B3	T	T	T
E 147v-B4	T	T	T
E 149-B4	T	T	F
E 149v-B4	T	F	F
E 151-B4	T	F	T
E 151v-B4	T	T	T
E 153-B4	T	T	F
E 153v-B4	T	T	T
E 155-B4	T	T	T
E 155v-B4	T	T	T
E 157-B4	T	T	T
E 157v-B4	T	T	T
E 159-B4	T	T	T
E 159v-B4	T	T	T
E 15-B4	T	T	T
E 15v-B4	T	T	T
E 161-B4	T	T	T
E 161v-B4	T	T	T
E 163-B3	T	T	T

E 163v-B3	T	T	T
E 165-B3	T	T	T
E 165-B3 not	T	T	T
E 165v-B4	T	T	T
E 167-B4	T	T	T
E 167v-B4	T	T	T
E 17-B5	T	T	T
E 19-B5	T	T	T
E 19v-B5	T	T	T
E 1-B5	T	T	F
E 1v-B5	T	T	T
E 21-B5	T	T	T
E 21v-B5	T	T	T
E 23-B5	T	T	T
E 23v-B5	T	T	T
E 24-B5	T	T	F
E 24v-B5	T	F	T
E 25-B5	T	T	T
E 25v-B5	T	T	T
E 27-B6	T	T	T
E 27v-B6	T	T	T
E 29-B6	T	T	T
E 29v-B6	T	T	T
E 31-B6	T	T	T
E 31v-B6	T	T	T
E 33-B6	T	T	T
E 35-B6	T	T	T
E 35v-B5	T	T	T
E 37-B5	T	T	T
E 39-B5	T	F	F
E 3-B6	T	T	T
E 3v-B6	T	T	T
E 41-B5	T	T	T
E 41v-B7	T	T	F
E 43-B7	T	T	T
E 43v-B7	T	T	T
E 45-B7	T	T	T
E 45v-B7	T	T	T
E 47-B7	T	T	T
E 47v-B7	T	T	T
E 49-B7	T	T	T
E 51-B6	T	T	T
E 51v-B6	T	T	F
E 53-B6	T	T	T

E 55-B6	T	T	T
E 55v-B6	T	T	F
E 57-B6	T	T	T
E 57v-B6	T	T	T
E 59-B6	T	T	T
E 59v-B8	T	T	T
E 5-B8	T	T	T
E 5v-B8	T	T	T
E 61-B8	T	F	T
E 61v-B8	T	F	T
E 63-B7	T	T	F
E 63v-B7	T	T	T
E 65-B7	T	T	T
E 65v-B7	T	F	F
E 67-B7	T	F	F
E 67v-B7	T	T	T
E 69-B7	T	T	T
E 69v-B7	T	F	T
E 71-B7	T	T	T
E 71v-B7	T	F	F
E 73-B7	T	T	F
E 73v-B8	T	T	T
E 75-B8	T	T	T
E 75v-B8	T	T	T
E 77-B8	T	T	T
E 77v-B8	T	T	T
E 79-B8	T	T	T
E 79v-B8	T	T	T
E 7-B8	T	T	T
E 7v-B8	T	T	T
E 81-B8	T	T	T
E 81v-B8	T	T	T
E 83-B8	T	T	T
E 83v-B8	T	F	T
E 85-B9	T	T	F
E 85v-B9	T	F	T
E 87-B9	T	T	T
E 87v-B9	T	T	T
E 89-B8	T	T	T
E 89v-B8	T	T	T
E 91-B9	T	T	F
E 91v-B9	T	F	F
E 93-B9	T	T	F
E 93v-B9	T	T	F

E 95-B9	T	T	F
E 95v-B9	T	T	F
E 97-B9	T	T	F
E 97v-B9	T	T	T
E 99-B9	T	T	T
E 99v-B10	T	F	F
E 9-B10	T	T	T
E 9v-B10	T	T	T

7.2 Yolov8 base architecture



model	d (depth_multiple)	w (width_multiple)	r (ratio)
n	0.33	0.25	2.0
s	0.33	0.50	2.0
m	0.67	0.75	1.5
l	1.00	1.00	1.0
x	1.00	1.25	1.0

Figure 7.1: Yolov8 architecture [9].

7.3 Re-ordering result given by the algorithm

Listing 7.1: This ordering result is derived from the bounding box information predicted by the Yolov8+CBAM.

```
Recto           Verso
['E 1-B5.jpg', 'E 1v-B5.jpg']
['E 3-B6.jpg', 'E 3v-B6.jpg']
['E 5-B8.jpg', 'E 5v-B8.jpg']
['E 9-B10.jpg', 'E 9v-B10.jpg']
['E 7-B8.jpg', 'E 7v-B8.jpg']
['E 11-B3.jpg', 'E 11v-B2.jpg']
['E 163-B3.jpg', 'E 163v-B3.jpg']
['E 81-B8.jpg', 'E 81v-B8.jpg']
```

```
['E 117-B1.jpg', 'E 117v-B1.jpg']
['E 115-B1.jpg', 'E 115v-B1.jpg']
['E 127-B2.jpg', 'E 127v-B2.jpg']
['E 129-B2.jpg', 'E 129v-B2.jpg']
['E 133-B2.jpg', 'E 133v-B3.jpg']
['E 131-B2.jpg', 'E 131v-B2.jpg']
['E 137-B3.jpg', 'E 137v-B3.jpg']
['E 139-B3.jpg', 'E 139v-B3.jpg']
['E 147-B3.jpg', 'E 147v-B4.jpg']
['E 149-B4.jpg', 'E 149v-B4.jpg']
['E 159-B4.jpg', 'E 159v-B4.jpg']
['E 123-B2.jpg', 'E 123v-B2.jpg']
['E 13-B3.jpg', 'E 13v-B3.jpg']
['E 15-B4.jpg', 'E 15v-B4.jpg']
['E 19-B5.jpg', 'E 19v-B5.jpg']
['E 21-B5.jpg', 'E 21v-B5.jpg']
['E 23-B5.jpg', 'E 23v-B5.jpg']
['E 121-B2.jpg', 'E 121v-B2.jpg']
['E 119-B1.jpg', 'E 119v-B2.jpg']
['E 111-B1.jpg', 'E 111v-B1.jpg']
['E 109-B2.jpg', 'E 109v-B1.jpg']
['E 135-B3.jpg', 'E 135v-B3.jpg']
['E 107-B2.jpg', 'E 107v-B2.jpg']
['E 113-B1.jpg', 'E 113v-B1.jpg']
['E 141-B3.jpg', 'E 141v-B3.jpg']
['E 143-B3.jpg', 'E 143v-B3.jpg']
['E 27-B6.jpg', 'E 27v-B6.jpg']
['E 73-B7.jpg', 'E 73v-B8.jpg']
['E 25-B5.jpg', 'E 25v-B5.jpg']
['E 75-B8.jpg', 'E 75v-B8.jpg']
['E 59-B6.jpg', 'E 59v-B8.jpg']
['E 125-B2.jpg', 'E 125v-B2.jpg']
['E 95-B9.jpg', 'E 95v-B9.jpg']
['E 93-B9.jpg', 'E 93v-B9.jpg']
['E 91-B9.jpg', 'E 91v-B9.jpg']
['E 97-B9.jpg', 'E 97v-B9.jpg']
['E 161-B4.jpg', 'E 161v-B4.jpg']
['E 101-B1.jpg', 'E 101v-B1.jpg']
['E 37-B5.jpg', 'E 37v-B5.jpg']
['E 105-B2.jpg', 'E 105v-B2.jpg']
```

```
['E 35-B6.jpg', 'E 35v-B5.jpg']
['E 79-B8.jpg', 'E 79v-B8.jpg']
['E 53-B6.jpg', 'E 53v-B6.jpg']
['E 49-B7.jpg', 'E 49v-B7.jpg']
['E 47-B7.jpg', 'E 47v-B7.jpg']
['E 45-B7.jpg', 'E 45v-B7.jpg']
['E 155-B4.jpg', 'E 155v-B4.jpg']
['E 63-B7.jpg', 'E 63v-B7.jpg']
['E 43-B7.jpg', 'E 43v-B7.jpg']
['E 51-B6.jpg', 'E 51v-B6.jpg']
['E 69-B7.jpg', 'E 69v-B7.jpg']
['E 77-B8.jpg', 'E 77v-B8.jpg']
['E 151-B4.jpg', 'E 151v-B4.jpg']
```

Bibliography

[1] D. Meuwissen, "Dissecting the errera-sketchbook: A reconstruction based on material evidence,"

[2] S. J. GUDLAUGSSON, "The errera-schetsboek and lucas van valckenborch," *Oud Holland*, vol. 74, pp. 118–138, 1959, ISSN: 0030672X, 18750176. [Online]. Available: `http://www.jstor.org/stable/42722992` (visited on 02/20/2024).

[3] E. Bernick, *Drawing connections: New discoveries regarding cesare da sesto's sketchbook*, Aug. 2019. [Online]. Available: `https://www.academia.edu/40198587/Drawing_Connections_New_Discoveries_Regarding_Cesare_da_Sestos_Sketchbook`.

[4] *Workshop practice in early netherlandish painting: Case studies from van eyck through gossart*, Jul. 2017. [Online]. Available: `https://www.brepols.net/products/IS-9782503566689-1`.

[5] L. Stratmann, *Color system*. [Online]. Available: `https://web.cs.uni-paderborn.de/cgvb/colormaster/web/`.

[6] B. A. Wandell, *Foundations of vision*. Sinauer Associates, 1995.

[7] U. Michelucci, "Advanced cnns and transfer learning," in Apress, 2019.

[8] H. Gong, H. Li, K. Xu, and Y. Zhang, "Object detection based on improved yolov3-tiny," in *2019 Chinese Automation Congress (CAC)*, 2019, pp. 3240–3245. DOI: `10.1109/CAC48633.2019.8996750`.

[9] Ultralytics, *Yolov8*, Mar. 2024. [Online]. Available: `https://docs.ultralytics.com/`.

[10] J. Cao, F. Han, Y. Wang, M. Wang, X. Zheng, and H. Gao, "A novel yolov5-based hybrid underwater target detection algorithm combining with cbam and ciou," in *2023 China Automation Congress (CAC)*, 2023, pp. 8060–8065. DOI: `10.1109/CAC59555.2023.10451391`.

[11] J. Kim and H.-S. Lim, "Neural network with binary cross entropy for antenna selection in massive mimo systems: Convolutional neural network versus fully connected network," *IEEE Access*, vol. 11, pp. 111 410–111 421, 2023. DOI: `10.1109/ACCESS.2023.3322679`.

[12] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[13] M. F. Rizal, R. Sarno, and S. I. Sabilla, "Canny edge and hough circle transformation for detecting computer answer sheets," in *2020 International Seminar on Application for Technology of Information and Communication (iSemantic)*, 2020, pp. 346–352. DOI: `10.1109/iSemantic50169.2020.9234208`.

[14] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," *ACM Trans. Graph.*, vol. 28, no. 3, p. 24, 2009.

[15] S. F. C. Haviana, D. Kurniadi, *et al.*, "Average hashing for perceptual image similarity in mobile phone application," *Journal of Telematics and Informatics*, vol. 4, no. 1, pp. 12–18, 2016.

[16] X.-m. Niu and Y.-h. Jiao, "An overview of perceptual hashing," *ACTA ELECTONICA SINICA*, vol. 36, no. 7, p. 1405, 2008.

[17] e. Dwyer. "Roboflow." (2024), [Online]. Available: `https://roboflow.com/`.

[18] M. Nawae, P. Maneelert, C. Choksuchat, T. Phairatana, and J. Jaruenpunyasak, *A comparative study of yolo models for sperm and impurity detection based on proposed augmentation in small dataset*, 2023. [Online]. Available: `https://ieeexplore-ieee-org.proxy.library.uu.nl/stamp/stamp.jsp?tp=&arnumber=10317746&tag=1`.

[19] B. Dayma, S. Patil, P. Cuenca, *et al.*, *Dall·e mini*, Jul. 2021. DOI: `10.5281/zenodo.5146400`. [Online]. Available: `https://github.com/borisdayma/dalle-mini`.

[20] H. Li, Dec. 2013. [Online]. Available: `https://www.researchgate.net/publication/261017148_Simulation_of_Ink_Diffusion_on_Xuan_Paper`.

[21] Q. Zhang, Y. Sato, J.-y. Takahashi, K. Muraoka, and N. Chiba, "Simple cellular automaton-based simulation of ink behaviour and its application to suibokuga-like 3d rendering of trees," *The Journal of Visualization and Computer Animation*, vol. 10, no. 1, pp. 27–37, 1999. DOI: `https://doi.org/10.1002/(SICI)1099-1778(199901/03)10:1<27::AID-VIS194>3.0.CO;2-C`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/%28SICI%291099-1778%28199901/03%2910%3A1%3C27%3A%3AAID-VIS194%3E3.0.CO%3B2-C`. [Online]. Available: `https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291099-1778%28199901/03%2910%3A1%3C27%3A%3AAID-VIS194%3E3.0.CO%3B2-C`.

[22] R. Dunlop. "Introduction to catmull-rom splines." (2005), [Online]. Available: `https://www.mvps.org/directx/articles/catmull/`.

[23] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, Springer, 2014, pp. 740–755.