University of Science and Technology of Hanoi

ResUNet-Based Comic Panel Extraction: A Hybrid Approach

Members

Nguyen Quang Huy - 22BI13195

Nguyen Tuan Khai - 22BI13202

Nguyen Hai Dang - 22BI13073

Le Linh Long - 22BI13262

Pham Thai Son - 22BI13397

Supervisor

Assoc. Prof. Tran Giang Son

Contents

List of Figures

List of Tables

Abstract

In comics, panels are considered as one of the main components as they contain elements such as characters, dialogs, etc. Being able to extract panels automatically is of great importance, as it allows for ease of viewing for users reading from small displays, such as phone and tablet screens. A number of methods have been proposed in recent years, using both image processing approach and deep learning approach. In this paper, we suggest a novel method that unites both of these paradigms, using ResUNet model for panel segmentation and a post-process pipeline to augment the results of more difficult pages. We tested on 2 datasets of different Japanese comics and achieved significant results.

Keywords— Comic, panel segmentation, image processing, ResUNet

1 Introduction

Artists have been exploring diverse mediums through which they can convey their stories for a long time, ranging from painting and composing to writing novels and producing films. Among these, comics - characterized as a sequence of pages composed of panels - have gained popularity in recent years, especially among adolescents. Since its inception, comics have been published in traditional formats similar to conventional books, often involving multiple volumes. However, with the rapid developments in technology we see today and the advent of digital platforms, a growing area of research has emerged focusing on the task of converting comics from physical to digital formats. This shift is relevant for enhancing the reading experience on electronic devices, particularly smartphones and tablets where reading a full comic page on a small display may cause some inconvenience to the viewer. Therefore, our research will emphasize the development of methods for automatic panel segmentation, specifically with the aim of overcoming the persistent challenges in this field involving comic series with variation in style and format.

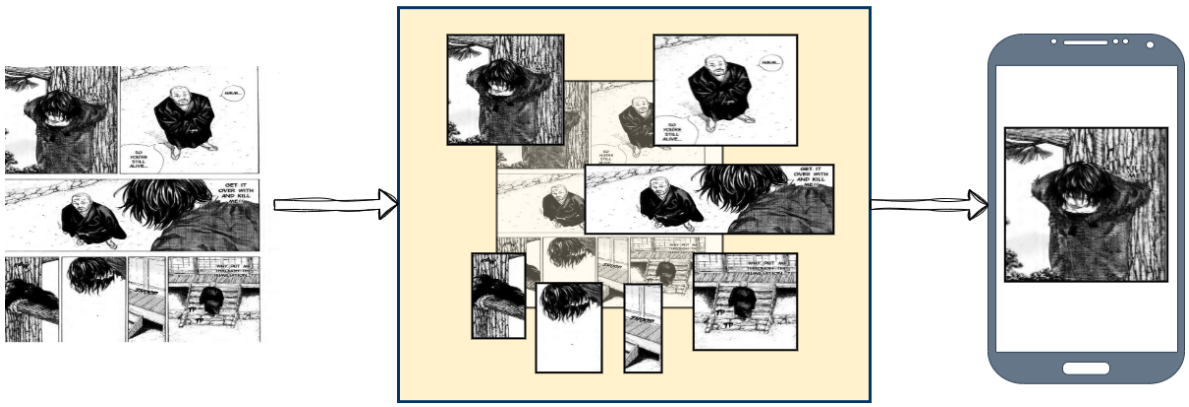


Figure 1: Vagabond Panel Extraction for displaying on small devices [11]

Panels can be defined as quadrilateral shapes containing different components making up a story: characters, text balloons, scenery, etc. They are often bounded by a black line and separated by "gutters", which are channels of white pixels that run between panels [4]. Figure 2 shows different types of panels that we can observe in a comic. A flat page (Figure 2a) is the simplest type, with rectangle panels separated by white gutters. Figure 2b is a variation containing non-rectangle panels. In more complicated cases, panels are merged (as in Figure 2c, 2g) by characters, balloons, etc, depending on the authors' creative decisions. Some panels may not have a bounding black line (Figure 2d), which can lead to difficulties in locating their boundaries. Pages can also be colored (Figure 2f) and have a combination of all mentioned components: extended character, text as in Figure 2h.

1.1 Image Processing Approach

From our findings, [1, 2] are the first papers presenting methods to accomplish the task of panel extraction. In their work , they employed the density gradient algorithm to detect white stripes between panels. However, this methodology was based on the assumption that the panels were filled black to begin with prior to the application of their algorithm, and it remains unclear whether this first step was done manually or automatically.

(a) Flat page

(b) Non-rectangular panels

(c) Overlapping elements

(d) Non-bounded panels

(e) Colored page

(f) Non-quadrilateral panels

(g) Extended text/Black panel background

(h) Combination

Figure 2: Examples of comic pages [27, 28, 29]

Connected Component Labeling CCL is an algorithm used to find all the white blobs inside a binary image. The algorithm was first introduced by [3], which performed perfectly on flat pages (i.e without any irregularities). However, for more complicated pages, more sophisticated techniques have to be utilized. [3] proposed a pipeline that detects many parts of the pages, e.g., division line, frame border, and comic balloons. [5] extracted all the bounding boxes of possible blobs and used K-means followed by a topology filter. A persistent flaw in both these methods is that they are not sufficient for non-rectangular panels. [9] first used CCL to fill all panels as black (including the extended objects) then applied a series of operations to find the best line to split each panel. However their description provided regarding their methods proved to be ambiguous as to how they handle diagonal gutters and non-bounded panels. [4] also tried to fill all panels, but using the Region Growing algorithm instead. After that, they use morphological operations like erosion and dilation to separate merged panels, however the specifics of their implementation of morphology remains unclear.

Line and polygon detection Besides CCL, [7] proposed a pipeline consisting of two parts: line segment detection and combination into polygons. This method can work well for flat pages and extended objects, but not for borderless panels as pointed out in their paper.

1.2 Deep Learning Approach

Advancements in Deep Learning - Convolutional Neural Networks With the advent of deep learning, and particularly Convolutional Neural Networks (CNNs), methods in this domain began to shift from traditional computer vision techniques to more data-driven approaches. CNNs provided a way to automatically learn features from images, eliminating the need for manually engineered features. This concept notably saw its first breakthroughs with the use of CNNs for the task of panel detection in comics [16]. Their model was able to handle irregular and complex layouts, detecting panels even when there were overlapping elements or abnormal shapes. This marked a significant shift from rule-based methods to deep learning, and in the same year Raj and Ghoss made another significant contribution to the field. They proposed an end-to-end model, based on the U-Net architecture, which allowed for accurate segmentation and pixel-level boundary detection [17]. This work was a step up from the previously established frontier of panel detection, and consolidated the shift towards deep learning-based methods, focusing on semantic segmentation.

Further Improvements with Instance Segmentation and Shift towards Hybrid Architectures In later years, more sophisticated models like Mask R-CNN were proposed to provide an instance-level segmentation, which allowed for individual panel segmentation and identification [18]. As segmentation tasks continued to gain complexity, more advanced and hybrid architectures also emerged combining the complementary strengths of existing architectures, allowing for better handling of both feature extraction and pixel-level segmentation. [23, 11, 24] were among the vanguard of researchers exploring U-Net-like architectures for panel segmentation, due to their ability to learn fine-grained details in pixel-level tasks making it suitable for comic panel extraction. Building upon these advancements, our work introduces a novel ResUNet architecture trained from scratch for comic panel semantic segmentation. Introduced in 2018 [25], the architecture combines the strengths of both the ResNet and U-Net in applications for remote sensing tasks, specifically for extracting roads from satellite imagery. By incorporating residual connections from ResNet, the extended U-Net architecture was able to engage in learning richer hierarchical features without suffering from vanishing gradients.

1.3 Remaining challenges

In spite of remarkable advancements within this area of research, some challenges still persists for many researchers today. Below is a list detailing some of the key unresolved issues:

Overlapping and Attaching Panels: Correctly identifying and separating attached panels still remains a prominent issue. "The most seen issue we have encountered is the attaching/overlapping panels. The model cannot separate all these cases in which we need to investigate more in detail to find a solution."[11]

False Positives: This can result from non-panel regions being incorrectly detected as panels, particularly in cases where stylistic elements are misclassified. As [3] noted, "The result frames are more than the frame scene number whereas some non-frame detected as a frame."

Dependency on Quadrilateral/Rectangular Assumptions: Most methods proposed by researchers are reliant on the assumption of quadrilateral or rectangular panels, which does not account for non-standard designs or artistic choices made by authors. "The scene frame of comics is defined as a quadrilateral frame composed of four straight lines including picture in its interior." [2]

Handling Margins and Partial Borders: Traditionally, blank margins and non-existent borders have been a large problem for automated methods, particularly with computer vision-based approaches which are

focused on detecting clearly defined black panel borders. As [7] notes in their experiment, "The recursive line division process still can't handle the blank margin very well."

Boundary Fragmentation: Fragmented borders are caused by overlapping elements or stylistic choices, and can often lead to incomplete detections. For instance, [9] explains, "Extended contents (e.g., speech balloons, characters, comics art) can overlap two frames or more, fragmenting the boundary."

Complex Layouts: This can be described as pages containing a variety of issues that can result in poor detection performance like irregular panel shapes, extended objects or intricate designs. [4] highlights, "The segmentation of this type of pages is very difficult. The shape of panels is totally irregular and many objects overlap the panels." Our method significantly improves results on such layouts.

Gaps in Detection with Adjoining Panels: Panels with shared borders also pose significant difficulties for existing methods. As [5] observes, "Text balloons extending over the panel borders usually produce gaps in the detections."

In the rest of this paper, the outline will be structured as follow: section 2 is the detail of our method and section 3 is about the experiments that we conducted.

2 Method

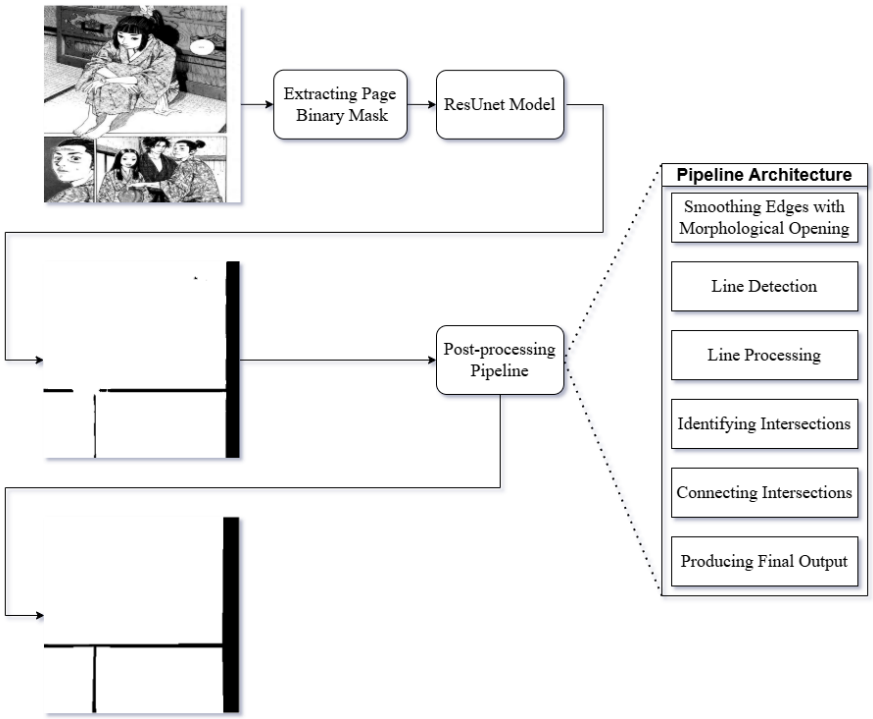


Figure 3: Overview of Our Panel Extraction Methodology

2.1 Model

In this study, we decided to consider this task as semantic segmentation or pixel classification, where each pixel can be classified as either panel or background. To this end, we propose a novel method using a ResUNet architecture that is trained from scratch. The ResUNnet model combines the ResNet structure (residual learning) with the U-Net structure (semantic segmentation), and the combination of these 2 architectures facilitates the training of deep representations without facing the vanishing gradients problem, a common issue when training deeper networks [12]. As we have established, comics dramatically differ from one another based on the artistic styles, flowing elements, and panel shapes used by the author, and can even vary based on screentone concentrations and decorative flourishes. Therefore, our model requires extensive training in order to generalize and adapt to these differences effectively, thus necessitating a way to mitigate the challenges of deeper networks. In addition to the use of ResUNet, one other distinguishing characteristic of our implementation, compared to existing methods proposed by researchers in this domain [11, 18], is that our model was trained entirely from scratch. This distinction is particularly critical because comics are a very domain-specific medium, and therefore the model to be trained from scratch in order to learn their nuances. The uniqueness in art styles

and features employed by artists often leads to pre-trained models, like the popular ImageNet-trained ResNet, to perform inadequately. Fine-tuning on domain-specific data remedies this problem by allowing the model to learn from the unique features of the comics, and can help bolster our results. Studies have now shown that training on domain-specific datasets significantly enhances performance in specialized tasks compared to relying on generic pre-trained models [13], as corroborated in the findings of [11]: *"the pre-trained weights from ImageNet classification task is obtained from natural images which are very different from comic images."*.

2.1.1 ResUNet architecture

The principal U-Net architecture can be divided into two halves: an encoder and a decoder that are linked through a bottleneck layer [14]. The encoder extracts deep features from the image input through residual blocks, while the decoder progressively reconstructs the image through upsampling and refinement of the segmentation map. ResUNet introduces residual connections in its encoder, enabling the preservation of both high-level features and fine-grained details for the image segmentation process.

Encoder The encoder progressively downsamples the input image while extracting features at various scales. Each stage of the encoder consists of a residual block, which incorporates skip connections to prevent the vanishing gradient problem and facilitate the learning of deeper representations. These residual blocks use two convolutional layers with batch normalization and ReLU activations, followed by max pooling to reduce spatial dimensions. The encoder has four stages, with the number of feature channels doubling at each stage (32, 64, 128, and 256 channels).

Bottleneck The bottleneck serves as the deepest layer in the network and bridges the encoder and decoder. It employs a residual block with 512 channels to capture the most abstract and high-level features of the input image.

Decoder The decoder upsamples the feature maps and fuses them with corresponding feature maps from the encoder through skip connections, allowing the model to retain spatial information lost during downsampling. Each decoder stage consists of a transpose convolution layer for upsampling, concatenation with encoder features, and a residual block to refine the segmentation output. The number of channels decreases progressively in the decoder (256, 128, 64, and 32 channels).

Output Layer The final layer applies a 1x1 convolution with a sigmoid activation function, producing a single-channel probability map representing the segmentation output.

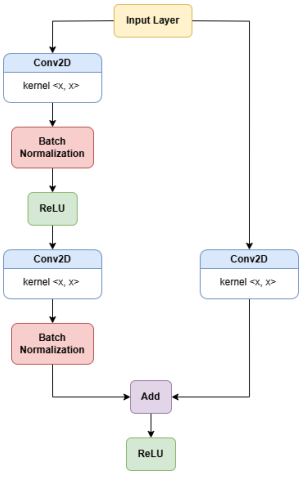


Figure 4: Residual Block

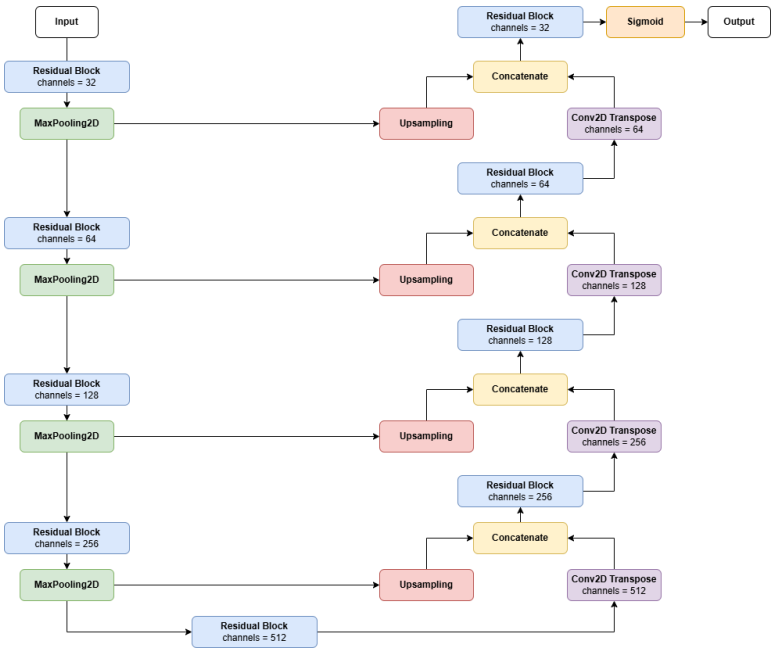


Figure 5: High-Level Representation of our ResUNet model Architecture

2.2 Post processing

The model ResUNet, while more adept at generalizing to unseen patterns in the images when compared to traditional computer vision methods, is still susceptible to outputs containing artifacts such as noise/irregular edges/disconnected regions. These issues can arise due to several challenges, especially in the context of comic books which are varied by nature. One of the most significant problem we encountered in our model was while the model could overcome small overlapping elements, larger elements still proved difficult to generalize and adapt to. In addition, panels containing noisy elements or sharp color transitions resembling their outer edges can result in spurious predictions and introduce noise within the panel regions, as seen in figure 6. To address these issues, we devised a post-processing pipeline, applying a series of operations to refine the model's output and enhance the structural coherence and interpretability of the segmentation results.

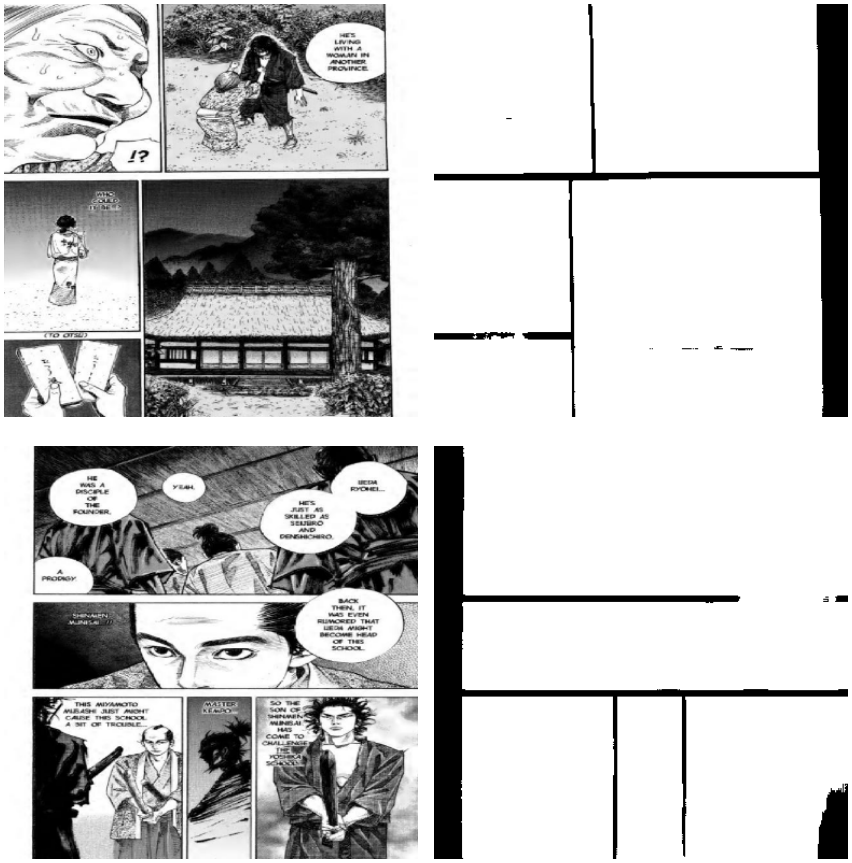


Figure 6: Example issues currently encountered by the ResUNet model, shown on Vagabond comic pages. The left-hand side shows the original page while the right shows the predicted output.

The following section will describe in detail each step in our tailored post-processing pipeline to augment the results of our model.

2.2.1 Smoothing Edges with Morphological Opening

In the first step, we address the irregularities in the segmented boundaries of these prediction masks by applying morphological opening, which is a composite operation of erosion followed by dilation. We define a structuring element B to process the input binary mask A, with the goal of removing small artifacts and smoothing rough lines. The morphological opening operation can be mathematically defined as:

$$A \circ B = (A \ominus B) \oplus B$$

where $\ominus$ and $\oplus$ denote erosion and dilation respectively.

$$(A \oplus B)(x, y) = \max\{A(x - i, y - j) \mid (i, j) \in B\}$$

$$(A \ominus B)(x, y) = \min\{A(x + i, y + j) \mid (i, j) \in B\}$$

where:

- (x, y): The coordinates of the pixel being operated on.
- (i, j): Coordinates of the structuring element B.
- |: Reads as 'such that', used to denote the condition that $(i, j) \in B$.

These operations systematically erode and rebuild boundaries resulting in cleaner, more uniform edges. Following this step, the binary mask undergoes some simple morphological processing, specifically the iterative applications of dilation and erosion operations. This process is to eliminate residual noise as well as connect the smaller gaps within the masks.

2.2.2 Line detection

The second step involves identifying linear structures in the processed binary mask corresponding to the potential panel boundaries on the comic page. This step involves applying Canny edge detection followed by Hough Transform to extract straight lines.

2.2.2.1 Canny Edge Detection Canny Edge detection is a multi-stage edge detection algorithm developed by John F.Canny in 1986. The process is comprised of 5 steps.

Noise Reduction To minimize the influence of noise, the image is smoothed with a Gaussian filter with a 5×5 kernel so that minor variations in intensity do not affect the edge detection process.

Gradient Computation The intensity gradients of the image are calculated using the Sobel operator, which approximates the first derivatives in the x- and y- directions. The Sobel kernels are defined as:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

For each pixel (i, j) in the image, the gradients $G_x(i, j)$ and $G_y(i, j)$ obtained by convolving the image I with these kernels. The gradient magnitude is computed as:

$$G(i, j) = \sqrt{G_x(i, j)^2 + G_y(i, j)^2}$$

The gradient direction $\theta(i, j)$ is calculated as the arctangent of the ratio of the G_y and G_x gradients:

$$\theta(i, j) = \text{atan2}(G_y(i, j), G_x(i, j))$$

This angle indicates the orientation of the gradient.

Non-Maximum Suppression To refine the edges detected, non-maximum suppression is applied. This step retains pixels with the largest gradient magnitude along the gradient direction while suppressing all others. The gradient direction is discretized to the nearest of the four main orientations: 0°, 45°, 90°, or 135°.

Hysteresis Thresholding Edges are classified based on their gradient magnitude into strong edges, weak edges and non-edges using two gradient magnitude thresholds (T_{high} and T_{low}):

$$\text{Edge(i, j)} = \begin{cases} \text{strong}, & \text{if } G(i, j) \geq T_{\text{high}}, \\ \text{weak}, & \text{if } T_{\text{low}} \leq G(i, j) < T_{\text{high}}, \\ \text{non-edge}, & \text{if } G(i, j) < T_{\text{low}}. \end{cases}$$

Weak edges connected to strong edges are preserved, while isolated weak edges are discarded, giving us the binary edge map representing the outline of the boundaries with which we can perform Hough Line Transformation.

2.2.2.2 Hough Line Transform

Next, Hough Transform is employed to identify straight lines in the edge-detected image. This technique converts the problem of detecting lines in Cartesian space to detecting peaks in a parameterized space, facilitating efficient line extraction and preparing us for an upcoming step. To understand the concept surrounding this technique, we first need to establish the Cartesian space, wherein a line is defined as:

$$y = mx + c$$

This representation is problematic for vertical lines, where the slope m is undefined. To overcome this, the Hough Transformation employs a polar representation:

$$\rho = x \cos\theta + y \sin\theta$$

where:

- ρ: The perpendicular distance from the origin to the line.
- θ: The angle between the perpendicular and the O_x axis.

In this parameter space, each edge pixel contributes to a sinusoidal curve in the accumulator array, and lines in the image correspond to the peaks as multiple edge pixels aligned on the same line vote for the same polar parameter space (ρ, θ).

In practice, we employed the Probabilistic Hough Transform [19], which retains the core concepts of the Hough Transform while emphasizing efficiency by randomly sampling a subset of edge pixels as opposed to all edge pixels. In addition, with this method, we are able to identify the line segments directly, represented by their endpoints (x_1, y_1) and (x_2, y_2), rather than infinite lines. To adapt the Probabilistic Hough Transform to the specifics of the edge-detected image, several configurable parameters govern its behavior. These parameters control aspects such as the threshold for detecting prominent lines, the minimum length of line segments to be considered valid, and the permissible gap between collinear segments to merge them into a single line.

2.2.3 Line Classification

Following the previous step of line detection, these lines are then grouped based on their orientation with respect to the Ox axis into two categories: horizontal and vertical. This step serves as the foundation for the next task of finding the intersection between these lines. To do this, we classify the orientation of each detected line segment using its angle θ in the polar representation (ρ, θ). For each line segment AB defined by endpoints (x_1, y_1) and (x_2, y_2), θ can be geometrically calculated as:

$$\theta = \arccos\left(\frac{\mathbf{AB} \cdot \mathbf{ox}}{\|\mathbf{AB}\| \cdot \|\mathbf{i}\|}\right)$$

where:

- $\mathbf{AB} = (x_2 - x_1, y_2 - y_1)$: The vector representing the line segment.
- $\mathbf{i} = (1, 0)$: The unit vector along the Ox axis.
- $\|\mathbf{AB}\| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$: The magnitude of $\mathbf{AB}$.

Segments where the angle θ is close to $90°$, indicating approximate perpendicularity to the O_x axis, are classified as vertical lines. Similarly, segments where θ is close to $0°$ or $180°$, are close to parallel with respect to the O_x axis and thus classified as horizontal lines. These angles are in a $0°$ or $180°$ range, and thresholds are applied to assign lines to their respective categories. For example, angles within $45° \leq \theta < 135°$ may be considered vertical, while other angles are treated as horizontal.

2.2.4 Noise Deduction

In this section, we define noise as lines that are outside the expected range of behaviors. As illustrated in Figure 7, the sample has 3 horizontal and 3 vertical lines when visually examined. But if analyzed more closely, we will find that every major line consists of smaller line segments. In the example image, we were able to detect 54 of these segments: 22 are horizontal lines, and 32 are vertical lines. Interestingly, those individual segments in each of the groups are not aligned in an optimal manner, as is the case within the horizontal line clusters where each line has an angular deviation of $0.1°$, $1°$, or $5°$ from the O_x axis rather than being perfectly aligned (i.e., having an angular deviation of $0°$). This angular difference adds some complexity to the next step of finding intersections, and thus an iterative line correction procedure is proposed to increase alignment and make the analysis easier.

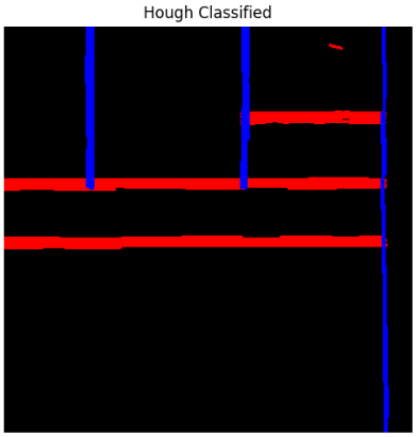


Figure 7: Segmentation mask after line classification. Segments of lines that slightly deviates from the angular position of eachother are present within each major line.

Line Grouping Detected lines are grouped based on their geometric properties, with the process conducted separately for vertical and horizontal lines. Assume that we have 2 line objects, line A and line B, defined as:

$$A(\theta_a, R_{a_1}, R_{a_2}, R_{a_3})$$
$$B(\theta_b, R_{b_1}, R_{b_2}, R_{b_3})$$

where:

- θ: The angle of the line with respect to reference (O_x) axis.
- R_1, R_2, R_3: The distance from the line to three different reference points in the image.

Reference points are chosen at random such that they do not align on the same line and are appropriately spaced. Preferably, these points form an acute triangle (i.e. all angles are less than 90°).

The difference between the distances and angles for the two lines are calculated as:

$$(|R_{a_1} - R_{b_1}|, |R_{a_2} - R_{b_2}|, |R_{a_3} - R_{b_3}|) = (D_1, D_2, D_3)$$

$$(|\theta_{a_1} - \theta_{b_1}|) = (\Delta\theta_1)$$

If any of these variables $(D_1, D_2, D_3, \Delta\theta)$ exceed a certain threshold (distance or angle difference threshold), the two lines are thus classified as separate groups.

Determining the General Angle Each line has an associated θ within its polar representation, and from the set of lines, we compute the mode of the corresponding θ's. If the mode can not be extracted (i.e. no dominant angle in the set), we will take the median of all θ's. Lines that deviate from the computed general angle are subsequently processed.

Line Rotation To ensure all lines within a set possess an uniform theta θ (i.e θ_{general}), we compute the angle of rotation as:

$$\hat{\theta} = \theta_{\text{general}} - \theta_{\text{line A}}$$

When rotating a point $P(x, y)$ in a Cartesian coordinate system xy through an angle of $\hat{\theta}$, the transformed $P(\hat{x}, \hat{y})$ in the new coordinate system $\hat{x}\hat{y}$ are described as:

$$\hat{x} = xcos\hat{\theta} + ysin\hat{\theta}$$

$$\hat{y} = -xsin\hat{\theta} + ycos\hat{\theta}$$

This transformation rotates the lines in a counterclockwise direction about the origin $O(0, 0)$, therefore to perform it effectively the origin $O(0, 0)$ must be aligned with the center point of the line. After rotation is performed, the line is shifted back to its original position.

2.2.5 Finding line intersections

With the lines now classified into categories of vertical and horizontal, we can find the intersection between those two groups. If we define two example lines A and B by their endpoints, as expressed in the following form:

$$A((x_{a_1}, y_{a_1}), (x_{a_2}, y_{a_2})), \quad B((x_{b_1}, y_{b_1}), (x_{b_2}, y_{b_2})).$$

The intersection point (p_x, p_y) between line A and line B can be computed using the following formulas :

$$p_x = \frac{(x_{a_1}y_{a_2} - y_{a_1}x_{a_2})(x_{b_1} - x_{b_2}) - (x_{a_1} - x_{a_2})(x_{b_1}y_{b_2} - y_{b_1}x_{b_2})}{(x_{a_1} - x_{a_2})(y_{b_1} - y_{b_2}) - (y_{a_1} - y_{a_2})(x_{b_1} - x_{b_2})},$$

$$p_y = \frac{(x_{a_1}y_{a_2} - y_{a_1}x_{a_2})(y_{b_1} - y_{b_2}) - (y_{a_1} - y_{a_2})(x_{b_1}y_{b_2} - y_{b_1}x_{b_2})}{(x_{a_1} - x_{a_2})(y_{b_1} - y_{b_2}) - (y_{a_1} - y_{a_2})(x_{b_1} - x_{b_2})}.$$

Here, p_x and p_y represent the coordinates of the intersection point between line A and line B. This process returns all line-to-line intersections, also referred to as "inner intersections". Additionally, we also compute the intersections that each line makes with the page boundaries. Since these lines extend towards infinity, each line would intersect the outer border of the comic page at two separate points, or border intersections.

2.2.6 Clustering intersection points

With the line intersections we've obtained, we apply clustering in order to group nearby points that likely belong to the same panel corner. This step is to make sure that redundant or tightly spaced groups of points are consolidated into singular centroids, which will allow for clearer line connection in the penultimate step. To this end we used the Mean-Shift Algorithm [20], a non-parametric iterative technique that identifies cluster centroids based on the local density of points, achieving similar results to K-Means but without having to specify the number of clusters. The algorithm works by shifting each point towards the point of maximum density as defined by a kernel, which can be mathematically written as :

$$f(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\|x - x_i\|}{h}\right)$$

where:

- h: Bandwidth parameter controlling neighborhood size.
- $K(u)$: Kernel function. In our experiment we used the flat (uniform) kernel $K(u) = \begin{cases} 1, & \text{if } \|u\| \leq h, \\ 0, & \text{otherwise.} \end{cases}$ with $u = \frac{x - x_i}{h}$. This assigns equal weight to all points within the bandwidth distance h from the current point x.
- $\|x - x_i\|$: Euclidean distance between x and x_i.
- d: Dimensionality of the data (here, $d = 2$).

From here, each point x is shifted iteratively based on the mean shift vector:

$$m(x) = \frac{\sum_{i=1}^{n} x_i K\left(\frac{\|x - x_i\|}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{\|x - x_i\|}{h}\right)} - x$$

The updated position of x after each iteration is:

$$x_{\text{new}} = x + m(x)$$

The algorithm iterates until the mean shift vector satisfies:

$$\|m(x)\| < \epsilon$$

where ϵ is the machine epsilon threshold for the smallest difference distinguishable between floating-point numbers (typically $\approx 2.22 \times 10^{-16}$ for `float64`). Upon convergence, x represents a cluster centroid.

2.2.7 Connecting Centroids

Prior to connecting any 2 given centroids, we define a custom metric to evaluate whether the connection would be valid. First, to compute the pixels between centroids, we use Bresenham's Line Algorithm [22] to determine the integer pixel coordinates approximating a straight line between two points (x_1, y_1) and (x_2, y_2).

Algorithm 1 Bresenham's Algorithm

Require: Two points (x_1, y_1) and (x_2, y_2)
Ensure: A list of points forming a line between (x_1, y_1) and (x_2, y_2)
  Initialization:
  Compute differences:
  $\Delta x \leftarrow |x_2 - x_1|$
  $\Delta y \leftarrow |y_2 - y_1|$
  Determine step directions:
  $s_x \leftarrow$ **if** $x_2 > x_1$ **then** 1 **else** -1
  $s_y \leftarrow$ **if** $y_2 > y_1$ **then** 1 **else** -1
  Init error term:
  Error $\leftarrow \Delta x - \Delta y$
  Set initial point:
  $x \leftarrow x_1$, $y \leftarrow y_1$
  Append (x, y) to the list of points
  while $(x, y) \neq (x_2, y_2)$ **do**
    Compute doubled error term:
    $e_2 \leftarrow 2 \times$ Error
    if $e_2 > -\Delta y$ **then**
      $x \leftarrow x + s_x$
      Error $\leftarrow$ Error $- \Delta y$
    end if
    if $e_2 < \Delta x$ **then**
      $y \leftarrow y + s_y$
      Error $\leftarrow$ Error $+ \Delta x$
    end if
    Append (x, y) to the list of points
  end while

To validate connections, each line segment is inspected by the number of colored pixels, corresponding to the vertical/horizontal boundary lines as opposed to the background, along its path. We established a threshold criterion where the connection is valid if:

$$\text{Valid Connection} = \frac{\text{Number of Significant Pixels}}{\text{Total Pixels}} \geq \text{Threshold}$$

Here, we use an arbitrary threshold x, meaning if more than x% of pixels between two centroids belong to a boundary line then we can verify the validity of their connection.

In the culmination of this post-processing pipeline, the overarching objective is to bridge the gaps between the panel boundaries that are open due to overlapping elements. Through this endeavor, we are thus able to augment the output of our ResUNet model significantly and overcome its largest challenges, as seen in Figure 9.

3 Experiment

3.1 Datasets

In our experiments, we combined data from 2 sources for the purposes of training our model and evaluating results. The first is a collection of 617 comic book pages and 3400 panels from the Vagabond series [27] that we hand-annotated ourselves, combined with a publicly-sourced dataset of mixed comics. This mixed dataset comprises 517 pages and 2220 panels, coming from Demon Slayer [28] and One Piece [29]. The variety in our selection of comic series ensures that we have a wide range of art styles, panel layouts, and visual complexities

from different artists so that our model would generalize panel structures across a broad spectrum of artistic expressions by authors.

Given the limited size of our datasets, data augmentation - including random flipping and rotation, as well as noise introduction, scaling, etc. - was employed to artificially expand our training data, thus allowing our model to better generalize across unseen data while alleviating the risk of overfitting. Post augmentation, the data for our models were divided as such:

	Training	Validation	Testing
Vagabond	698	73	73
Mixed	903	50	44

3.2 Model Implementation

3.2.1 Loss function and Metrics

In previous works that saw the deployment of deep learning architectures to the task of comic panel segmentation, authors often used a Binary Cross-Entropy (BCE) Loss with Sigmoid activation function. The BCE loss, defined as:

$$\text{BCE Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$$

is a pixel-wise classification loss that treats the segmentation task as a collection of independent binary classification problems. This approach is fundamentally flawed in situations where there exists a significant class imbalance, i.e. where either the panels or background regions occupy a larger fraction of the entire image. In such cases, models trained may exhibit bias toward predicting the majority class at the expense of minority class accuracy.

To address these challenges, we employed the Dice Loss function, which is particularly well-suited for handling class imbalances in segmentation tasks. Our dice loss is derived from the dice coefficient, a region-based similarity measure that directly optimizes the overlap between the predicted and ground truth segmentation masks.

$$\text{Dice Loss} = 1 - \frac{2 \cdot \sum(y_{\text{true}} \cap y_{\text{pred}}) + \epsilon}{\sum y_{\text{true}} + \sum y_{\text{pred}} + \epsilon}$$

where:

- y_{true} is the ground truth segmentation mask, and y_{pred} the predicted mask.
- ϵ is a small smoothing constant (10^{-6} in this context) to avoid division by zero.

Unlike BCE loss, Dice Loss focuses on the global overlap between regions, and directly penalizes misalignments of these regions. This property ensures that smaller regions are adequately learned during training, especially in cases where there are small/irregularly shaped panels or noisy borders. In addition to the loss function, we implemented the Mean Dice Coefficient as a custom evaluation metric, calculated as:

$$\text{Dice Coeff} = \frac{2 \cdot \sum(y_{\text{true}} \cap y_{\text{pred}}) + \epsilon}{\sum y_{\text{true}} + \sum y_{\text{pred}} + \epsilon}$$

and the mean value is calculated across all predictions in a batch. This metric adheres closer to the optimization objective for segmentation purposes.

3.2.2 Optimizer and Learning Rate Scheduling

We trained our model using the popular Adam optimizer, with a learning rate scheduler function to prevent overfitting at later epochs and allow the model to explore the data more liberally. Specifically, we implemented a cosine learning rate decay with a warm-up schedule to control the training process, as a constant or poorly tuned learning rate may lead to slow convergence, overshooting, or failure to escape local minima. Our approach combines:

- Linear Warm-Up: Gradually increasing the learning rate from 0 to $\eta_{\max}$ over a predefined number of warm-up steps T_{warmup}. This stabilizes the model during the initial training phase and avoids large gradient updates that could negatively affect the learning process
- Cosine Decay: Once the warm-up phase completes, the learning rate decays smoothly following a cosine function, allowing the optimizer to update model parameters more gradually.

$$\eta(t) = \begin{cases} \eta_{\max} \cdot \frac{t}{T_{\text{warmup}}} & \text{if } t < T_{\text{warmup}}, \\ \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left(1 + \cos\left(\frac{t - T_{\text{warmup}}}{T_{\text{decay}}} \pi\right)\right) & \text{otherwise.} \end{cases}$$

where:

- $\eta(t)$ is learning rate at step t.
- $\eta_{\max}$ is the target learning rate after warm-up phase.
- $\eta_{\min}$ is the minimum learning rate after decay cycle.
- T_{warmup} is the number of warm-up steps.
- T_{decay} is the number of decay steps.

This approach was inspired by SGDR (Stochastic Gradient Descent with Warm Restarts) proposed by Loshchilov and Hutter (2017) [26]. The cosine decay ensures a smooth reduction in the learning rate, which helps to prevent oscillations around local minima and allows for fine-tuning during the later stages of training.

3.2.3 Callbacks for Training

We implemented several callback mechanics during the training process, which was conducted over 150 epochs. This allowed us to manage the learning dynamics and preserve the best-performing model based on its performance on the validation set:

- *ModelCheckpoint*(): Monitored the validation Dice Loss after each epoch and saved the model with the lowest loss for later testing. The results we show in this research will be from these models as opposed to the ones trained to completion over 150 epochs.
- *ReduceLROnPlateau*(): Adaptively reduced the learning rate if the validation loss did not improve for a specified number of epochs.

3.3 Post-Processing

Separate from the main post-processing pipeline (demonstrated in Figure 9), we also perform an operation on the raw predicted binary mask to remove artifacts that can appear in segmentation masks, such as small background regions that occupy the predicted panels. We perform a connected component analysis and filtering operation, which finds the components of the panel that are disconnected based on geometric criteria and remove them from the panel, allowing us to refine our results further before connecting the panel boundary.

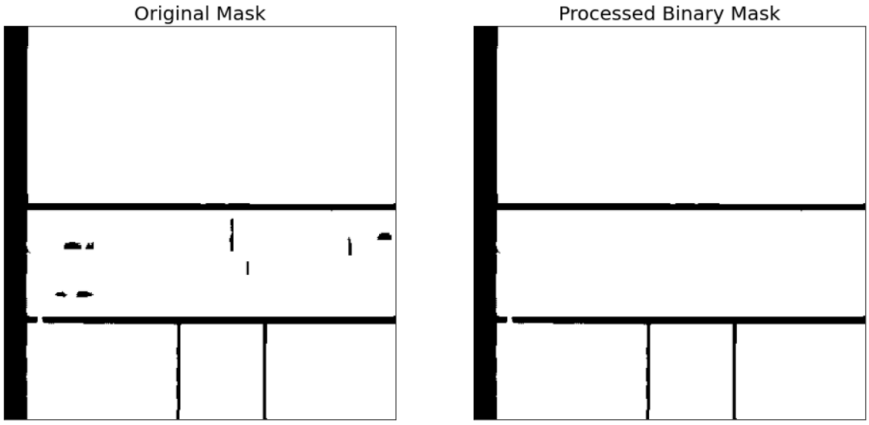


Figure 8: Before and after noise removal

Connected Component Labeling At first, we apply CCL to label the disjoint area of the binary mask. For further processing, each region is given a unique label (see Algorithm 2 in Section 3.4.1 for detailed implementation [2]).

Geometric Filtering For each connected component C_k, two geometric properties are computed:

- **Area,** $A(C_k)$: The total number of pixels in C_k.
- **Perimeter,** $P(C_k)$: The arc length of the boundary of C_k.

A connected component C_k is retained only if it satisfies the following criteria:

$$A(C_k) > T_A \quad \text{and} \quad P(C_k) > T_P,$$

where T_A and T_P are predefined thresholds for area and perimeter, respectively. The set of valid components is given by:

$$\mathcal{K} = \{k \mid A(C_k) > T_A \text{ and } P(C_k) > T_P\}.$$

Binary Mask Reconstruction Using the filtered components, a binary mask M is created as follows:

$$M(x, y) = \begin{cases} 1, & \text{if } (x, y) \in C_k \text{ for any } k \in \mathcal{K}, \\ 0, & \text{otherwise.} \end{cases}$$

The expected output is a cleaned binary segmentation mask M where we preserve only relevant components to be considered in subsequent steps.

After the implementation of our post-processing pipeline, a new critical issue arose where the newly connected lines introduced small, unintended foreground regions within panel boundaries or background areas. To address this problem, we apply a similar logic operation to the noise removal function earlier, whereby these artifacts are eliminated based on a threshold we set for the area and perimeter of each element.

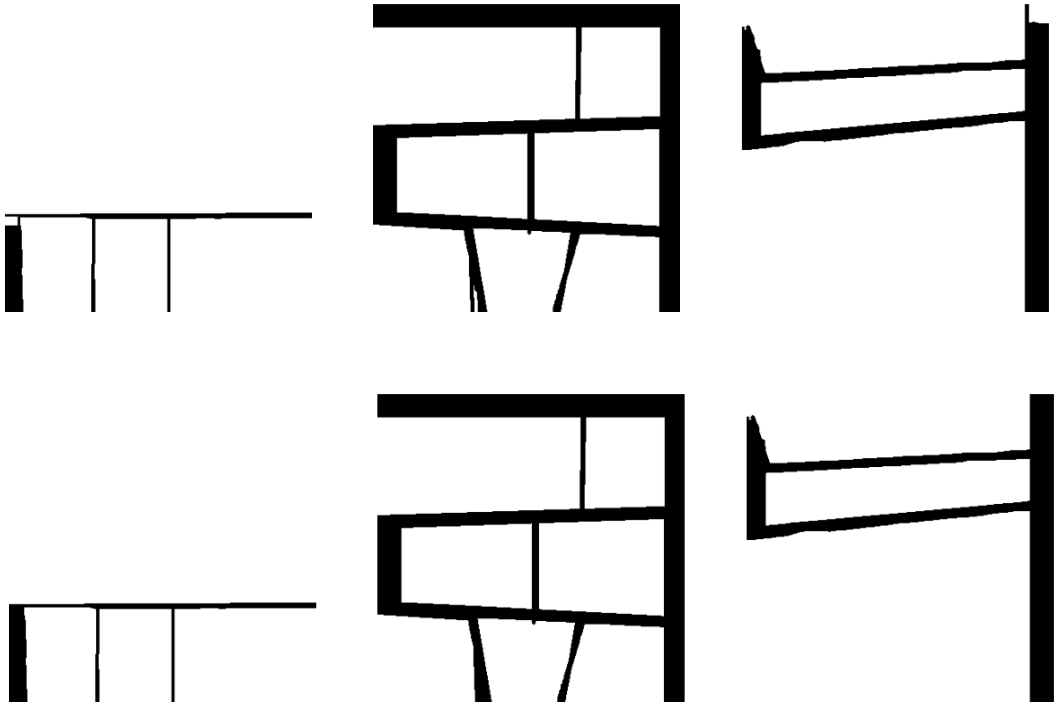


Figure 10: Before and After Removing Small Gaps

3.4 Evaluating results

We assess the results of our method by employing two distinct types of metrics - Intersection over Union and Dice Coefficient to evaluate segmentation results and page/panel accuracy to measure the model's overall proficiency in detecting and correctly identifying panels within the comic page. We will explain in more detail how each of these evaluation metrics are implemented, as well as present our results side by side with other researchers' proposed method within the domain to give a holistic view of our performance.

3.4.1 Segmentation-Specific Metrics

Both the Mean Intersection-over-Union (mIoU) and the Mean Dice Coefficient (mDice) are computed at the instance level by considering each individual segmented object within the binary mask. So, the first step before we can evaluate these metrics is to localize and isolate individual panels within the binary mask. We accomplish this by using the Connected Component Labeling (CCL) algorithm, which identifies and labels

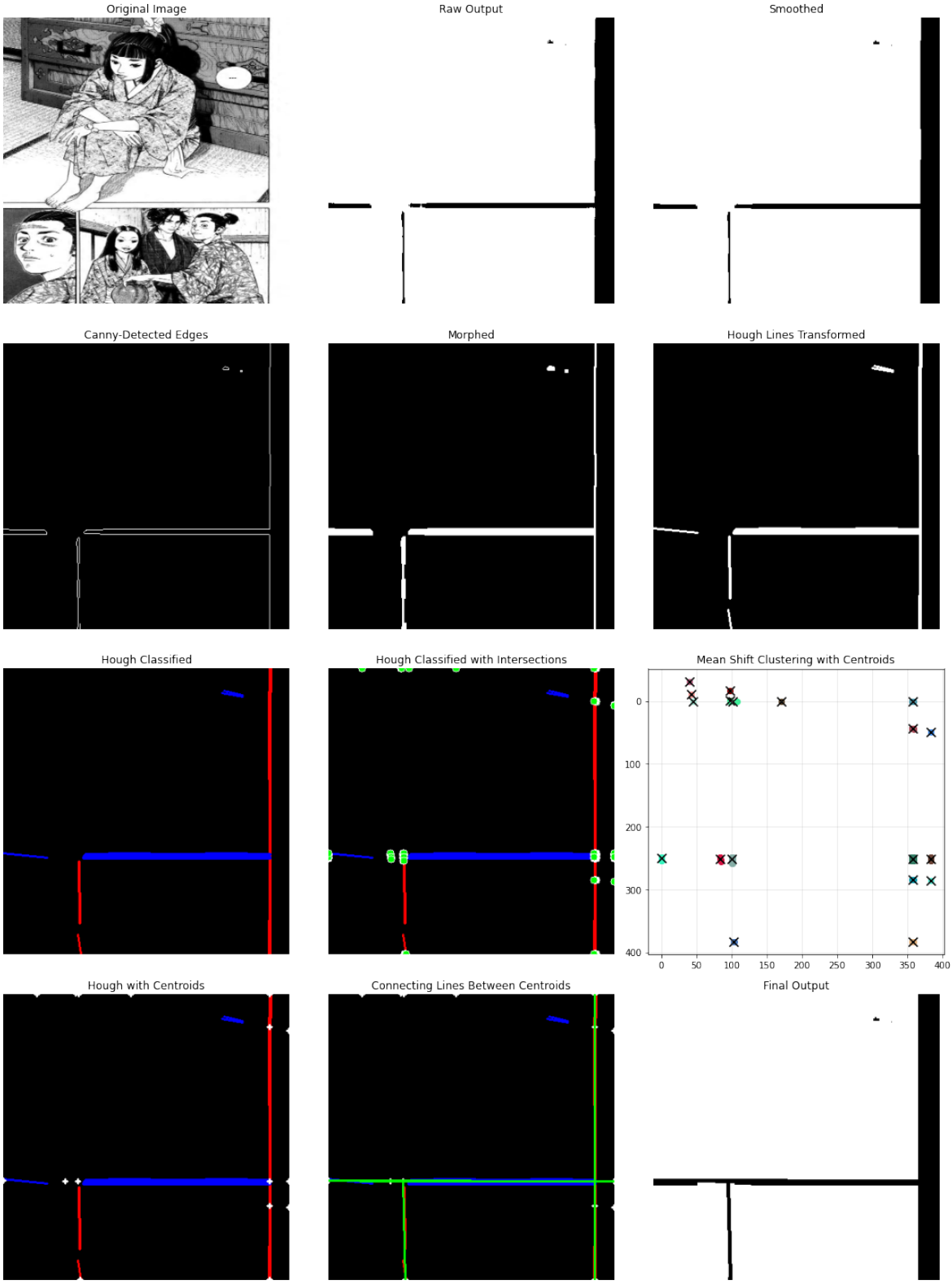


Figure 9: Full Post-Processing pipeline represented in iterative steps

distinct connected regions of pixels in the mask. Given a binary mask M of dimension $H \times W$, where each pixel $M(i,j) \in \{0,1\}$ represents either the background ($M = 0$) or the foreground ($M = 1$), our goal with CCL is to assign a unique label L_k to each connected region of foreground pixels (panels).

Two pixels $M(i,j)$ and $M'(i',j')$ are considered connected if:

$$\text{dist}((i,j),(i',j')) \leq 1 \quad \text{and} \quad M(i,j) = M'(i',j') = 1$$

where $\text{dist}((i,j),(i',j'))$ is the Manhattan distance. Our algorithm uses 8-connectivity, meaning they would define neighborhood relationships to include immediate horizontal and vertical neighbors as well as diagonal neighbors. Under 8-connectivity, two pixels (i,j) and (i',j') are connected if:

$$\max(|i-i'|, |j-j'|) \leq 1 \quad \text{and} \quad M(i,j) = M(i',j') = 1.$$

The CCL algorithm proceeds as follows:

Algorithm 2 Connected Component Labeling (CCL)

Require: Binary mask M of size $H \times W$
Ensure: Labeled mask L of size $H \times W$
  Init $L(i,j) \leftarrow 0$ for all (i,j)
  Init next_label $\leftarrow 1$
  Init table equivalence
  for each pixel (i,j) in M **do**
    if $M(i,j) = 1$ **then**
      neighbor_labels $\leftarrow$ Labels of neighbors in L
      if neighbor_labels $= \emptyset$ **then**
        $L(i,j) \leftarrow$ next_label
        equivalence[next_label] $\leftarrow$ next_label
        Increment next_label
      else
        $L(i,j) \leftarrow \min$(neighbor_labels)
        $l \in$ neighbor_labels
        for each l **do**
          equivalence[l] $\leftarrow \min$(equivalence[l], $L(i,j)$)
        end for
      end if
    end if
  end for
  for each pixel (i,j) in L **do**
    if $L(i,j) > 0$ **then**
      $L(i,j) \leftarrow$ equivalence[$L(i,j)$]
    end if
  end for
    return L

Mathematically, the output of the CCL algorithm is a labeled mask L such that:

$$L(i,j) = \begin{cases} k, & \text{if } M(i,j) \text{ is connected to the } k\text{-th object,} \\ 0, & \text{if } M(i,j) = 0. \end{cases}$$

This labeled mask L allows us to precisely identify and compute our evaluation metrics over individual panel instances both in the ground truth and the predicted masks.

Mean Intersection-over-Union (mIoU) Mathematically, the IoU for a given pair of masks is defined as:

$$\text{IoU} = \frac{|y_{\text{true}} \cap y_{\text{pred}}|}{|y_{\text{true}} \cup y_{\text{pred}}|}$$

where y_{true} and y_{pred} are the sets of pixels belonging to the predicted and the ground truth masks respectively. For each ground truth panel, the highest IoU value amongst the predicted panels is recorded, and we consider

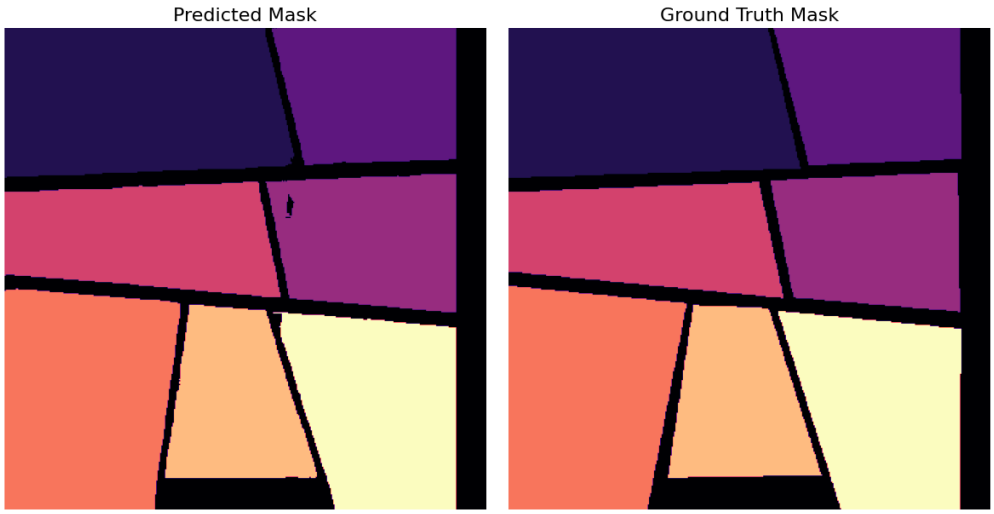


Figure 11: Labeled Predicted and Ground Truth masks

these two panels to be a pair. These individual IoU values are then averaged across all ground truth panels, giving us the mean IoU:

$$\text{mIoU} = \frac{1}{N} \sum_{i=1}^{N} \text{IoU}_i$$

where N is the total number of ground truth panels.

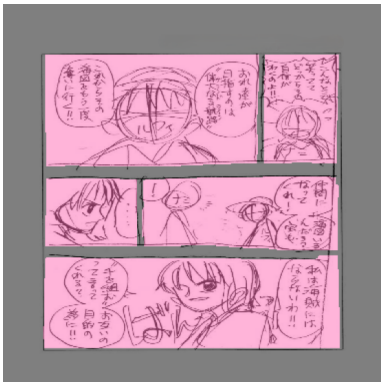
Mean Dice Coefficient (mDice) We also calculate the Dice Coefficient over each individual panel and the mean Dice Coefficient spanning each entire page, for which the mathematical definition has been given above.

3.4.2 Panel-wise Metrics

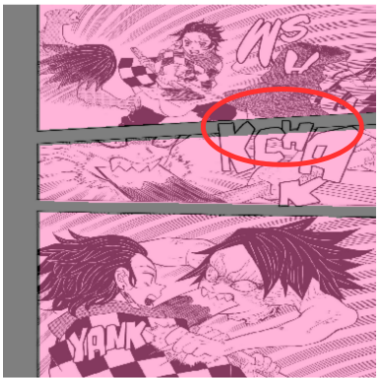
In addition to the segmentation metrics we provided earlier, we also aim to implement a panel-wise accuracy metric, which would allow for a more intuitive understanding of how well our method performs. While many papers have implemented this metric in order to evaluate and cross-evaluate amongst one another, there is no consensus as to what would constitute an "accurate" prediction of a panel. There are many dissenting opinions and proposals from scholars, ranging from counting by hand and visually examining the panels' overall shape to implementing a more algorithmic pixel-level accuracy measure.

Given the lack of agreement in this regard, we also propose our own method for obtaining panel-wise accuracy, inspired by Rigaud et al.'s suggestion [11]. Since we have obtained the Dice Coefficient of the individual panel regions in the previous section, we simply define a threshold $t = 0.8$, wherein if the Dice Coefficient between a true panel segment and its corresponding predicted segment exceeds this threshold we consider this to be an accurate prediction. This will give us a structured and unbiased method to obtain our accuracy, ensuring consistency throughout the evaluation. We also compute panel-wise Precision, Recall, and F1-Score to give a more comprehensive understanding of our model performance. As described before, our method presented some challenges like background artifacts in panel regions or gaps between the panel boundaries, which we proposed solutions for. These metrics can help us better gauge the extent to which we have been able to get rid of spurious predictions, and are understood as follows:

- **Precision:** The ratio of the number of correctly predicted panels to the number of panel regions predicted.
- **Recall:** Ratio of correctly predicted panels versus the number of correctly predicted panels and incorrectly predicted background regions.
- **F1-score:** Harmonic mean of precision and recall.

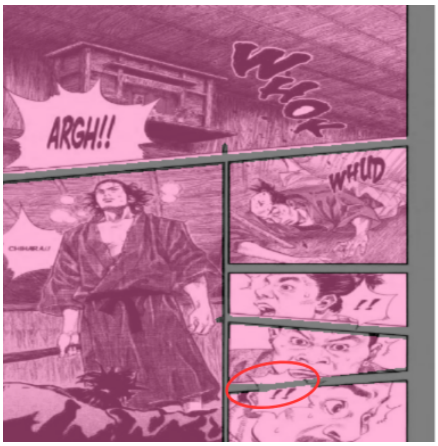


(a) Non-quadrilateral Panels



(b) Overlapping Elements

(c) Overlapping/Noisy Elements



(d) Overlapping Elements

(e) Colored Page/Overlapping Elements

Figure 12: Sample results from our model. [27, 28, 29]

	Vagabond		Mixed	
	No Post-Process	Post-Processed	No Post-Process	Post-Processed
Mean IoU	0.8929	0.9524	0.8843	0.9337
Mean Dice Coeff	0.9201	0.9688	0.9152	0.9588
Panel Accuracy	0.8846	0.9515	0.8775	0.9702
Page Accuracy	0.7031	0.8689	0.7674	0.9524
Precision	0.8638	0.9542	0.9033	0.9884
Recall	0.8658	0.9515	0.8648	0.9709
F1-Score	0.8559	0.9502	0.8743	0.9767

Table 1: Evaluation Results

3.5 Remaining issues

While our method was able to overcome several significant challenges for traditional panel extraction techniques, some problems still remain. With the logic we implemented in the intersection connection step of our post-processing pipeline, one particular issue arose where points that connect from border intersections to line intersection travels along existing boundaries. In these cases, the number of pixels belonging to a boundary exceed the threshold, and thus erroneous connections are made that in some cases may lead to unintended separations.

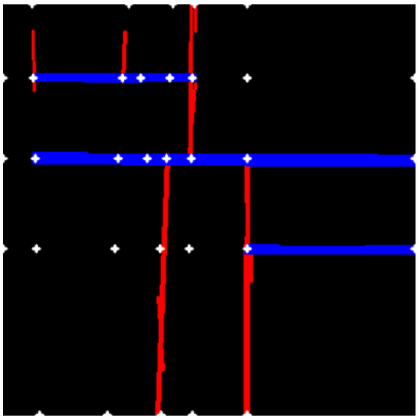


Figure 13: Classified Hough Lines with Centroid

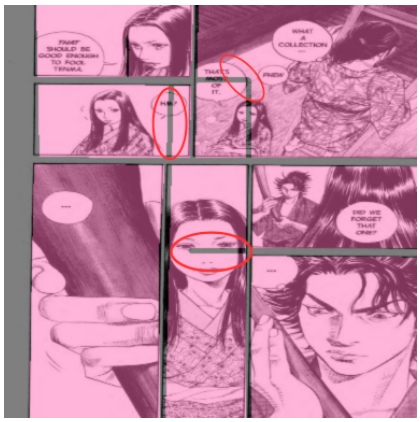


Figure 14: Sample Result

Furthermore, in cases of directly attached panels, our model still fails to separate them. This is a problem that persists in nearly all research on comic panel extraction — directly attached panels with no clear gutters or borders to separate them. Although segmentation methods have progressed, the reliable disassembly of attached panels has largely not been solved in the literature.

Figure 15: Segmentation of Pages with Attached/Overlapping Panels

Methods	**Metric**						
	IoU	**Dice**	**Panel Accuracy**	**Page Accuracy**	**Precision**	**Recall**	**F1**
[10] MLP-based X-Y recursive cut + projection profile verification	--	--	--	--	87.4%	93.53%	90.36%
[11] U-Net-based model with morphology.	--	--	--	--	82.15% (eBDtheque), 86.94% (Sequencity612)	78.47% (eBDtheque), 82.04% (Sequencity612)	80.27% (eBDtheque), 84.42% (Sequencity612)
[4] Region growing with morphology.	--	--	87.3%	64.3%	--	--	--
[6] Image processing and ontology-based inference	--	--	--	--	86.55%	81.24%	83.81%
[5] Connected component labeling + k-means, binarization + filtering	--	--	88.2%	66.7%	--	--	--
[21] Hough Transform and bounding box-based line labeling.	--	--	--	--	89%	82%	85%
Ours	**95.24% (Vagabond), 93.37% (Mixed)**	**96.88% (Vagabond), 95.88% (Mixed)**	**95.15% (Vagabond), 97.02% (Mixed)**	**86.89% (Vagabond), 95.24% (Mixed)**	**95.42% (Vagabond), 98.84% (Mixed)**	**95.15% (Vagabond), 97.09% (Mixed)**	**95.02% (Vagabond), 97.67% (Mixed)**

Table 2: Result Cross Evaluation with other methods

4 Conclusion

In this study, we proposed a novel approach for comic panel extraction by employing the ResUNet architecture which, when augmented with a strong post-processing pipeline, demonstrated outstanding performance with respect to panel segmentation and identification. Our methodology tackled several challenges such as varying artistic styles and complicated layouts, and showed improvements from traditional and earlier deep learning approaches.

Challenges still exist nonetheless. Panels with attached boundaries or very small gutters still proved difficult to resolve, as our model was not able to generalize to them and our post-processing pipeline could not create any meaningful connection without compromising the results of other predictions. Moreover, although our post-processing pipeline enhanced structural coherence, it caused problems like false connections, dividing panels that should exist without boundaries.

Future works should aim at optimizing the pipeline in order to prevent these errors and investigate architectures or multimodal methods that allow improved generalization. Despite these setbacks, our work presents a substantial step towards bridging the gap between traditional comics with their digital counterparts.

References

[1] Takamasa Tanaka, Kenji Shoji, Fubito Toyama, and Juichi Miyamichi. Layout analysis of scene frames in comic images. In Proceedings of The 13th International Display Workshops (IDW'06), VHFp - 20, December 6-8 2006.

[2] T. Tanaka, "Layout Analysis of Tree-Structured Scene Frames in Comic Images," 2007.

[3] K. Arai and T. Herman, "Method for Automatic E-Comic Scene Frame Extraction for Reading Comic on Mobile Devices," in 2010 Seventh International Conference on Information Technology: New Generations, Las Vegas, NV, USA: IEEE, 2010, pp. 370–375. doi: 10.1109/ITNG.2010.22.

[4] A. K. NGO HO, J.-C. Burie, and J.-M. Ogier, "Comics page structure analysis based on automatic panel extraction," Sep. 2011.

[5] C. Rigaud, N. Tsopze, J.-C. Burie, and J.-M. Ogier, "Robust Frame and Text Extraction from Comic Books," in Graphics Recognition. New Trends and Challenges, vol. 7423, Y.-B. Kwon and J.-M. Ogier, Eds., in Lecture Notes in Computer Science, vol. 7423. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 129–138. doi: 10.1007/978-3-642-36824-0

[6] C. Rigaud, "Knowledge-Driven Understanding of Images in Comic Books," in 2015 International Conference on Document Analysis and Recognition Workshops (ICDARW), Nancy, France: IEEE, 2015, pp. 444–451.

[7] L. Li, Y. Wang, Z. Tang, and L. Gao, "Automatic comic page segmentation based on polygon detection," Multimed Tools Appl, vol. 69, no. 1, pp. 171–197, Mar. 2014, doi: 10.1007/s11042-012-1241-7.

[8] Y. Wang, Y. Zhou, and Z. Tang, "Comic Frame Extraction via Line Segments Combination," *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 856–860, 2015.

[9] X. Pang, Y. Cao, R. W. H. Lau, and A. B. Chan, "A Robust Panel Extraction Method for Manga," in Proceedings of the 22nd ACM international conference on Multimedia, Orlando Florida USA: ACM, Nov. 2014, pp. 1125–1128. doi: 10.1145/2647868.2654990.

[10] E. Han, K. Kim, H. Yang, and K. Jung, "Frame Segmentation Used MLP-Based X-Y Recursive for Mobile Cartoon Content," in Human-Computer Interaction. HCI Intelligent Multimodal Interaction Environments, J. A. Jacko, Ed., Berlin, Heidelberg: Springer, 2007, pp. 872–881. doi: 10.1007/978-3-540-73110-8_96.

[11] V. Nguyen Nhu, C. Rigaud, and J.-C. Burie, "What do We Expect from Comic Panel Extraction?," in 2019 International Conference on Document Analysis and Recognition Workshops (ICDARW), Sydney, Australia: IEEE, Sep. 2019, pp. 44–49. doi: 10.1109/ICDARW.2019.00013.

[12] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[13] Tajbakhsh, N., Shin, J. Y., Gurudu, S. R., Hurst, R. T., Kendall, C. B., Gotway, M. B., & Liang, J. (2016). Convolutional neural networks for medical image analysis: Full training or fine tuning?. *IEEE Transactions on Medical Imaging*, 35(5), 1299-1312.

[14] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*.

[15] Agarwal, M., Lyu, D., Kim, H. S., & Lee, Y. S. (2019). Comic book layout analysis using convolutional neural networks. *Pattern Recognition*, 90, 93-104.

[16] Zhang, S., Chen, W., Song, Y., & Xia, L. (2018). Panel detection in comics using convolutional neural networks. *International Conference on Pattern Recognition*.

[17] Raj, R. C., & Ghosh, P. K. (2018). End-to-end comic strip panel extraction with deep learning. *Computer Vision and Image Understanding*.

[18] Rashid, M. H., Ahmad, M. Q., & Hussain, S. S. (2019). Panel extraction in comics using Mask R-CNN. *ICDAR 2019*.

[19] Kiryati, N., Eldar, Y., & Bruckstein, A. M. (1991). A probabilistic Hough transform. *Pattern Recognition*, *24*(4), 303–316.

[20] Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *17*(8), 790–799.

[21] R. Onishi, K. Tanaka, H. Sawano, Y. Suzuki, and S. Hotta, "Method for Creating Motion Comic from Printed Comic," in 2020 Nicograph International (NicoInt), 2020, pp. 51–54. doi: 10.1109/NicoInt50878.2020.00017.

[22] Bresenham, J. E. (1965). Algorithm for computer control of a digital plotter. *IBM Systems Journal*, *4*(1), 25–30.

[23] Kim, J. S., Kim, S. H., & Lee, J. Y. (2017). Deep comic layout analysis and generation with neural networks. *International Journal of Computer Vision*.

[24] Fernandez, M. J., Garcia, M., & Benassi, F. M. (2020). Comic layout analysis and generation with neural networks. *Computers & Graphics*.

[25] Zhang, Z., Yang, L., & Zheng, Y. (2018). Road Extraction by Deep Residual U-Net. *IEEE Geoscience and Remote Sensing Letters*, 15(5), 749–753.

[26] I. Loshchilov and F. Hutter, *SGDR: Stochastic Gradient Descent with Warm Restarts*, International Conference on Learning Representations (ICLR), 2017.

[27] Takehiko Inoue, Vagabond

[28] Koyoharu Gotouge, Demon Slayer

[29] Eiichiro Oda, One Piece