UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

Group Project Report

Bubble Extraction and Dialog Translation for Japanese Manga

Supervisor	Assoc. Prof. Tran Giang Son	
Group	49	
Students	Nguyen Lam Tung	23BI14446
	Nguyen Vu Hong Ngoc	23BI14345
	Hoang Khanh Dong	22BA13072
	Le Chi Thanh Lam	23BI14248
	Pham Quang Vinh	23BI14455
	Pham Quang Minh	23BI14296

List of Abbreviations

BERT	Bidirectional Encoder Representations from Transformers
BLEU	Bilingual Evaluation Understudy
CER	Character Error Rate
CNN	Convolutional Neural Network
COCO	Common Objects in Context
COMET	Crosslingual Optimized Metric for Evaluation of Translation
ELAN	Efficient Layer Aggregation Networks
IoU	Intersection over Union
JSON	JavaScript Object Notation
mAP	mean Average Precision
MT	Machine Translation
NLP	Natural Language Processing
OCR	Optical Character Recognition
UC	Use Case

Contents

Abstract

Manga translation is a multimodal task that requires the integration of visual understanding and natural language processing, as textual content is tightly coupled with visual layouts, speech bubbles, and artistic styles. Unlike standard document translation, manga translation involves complex page structures, non-linear reading orders, and text embedded within illustrations, posing significant challenges for full automation. To support manual manga translation, we present an end-to-end pipeline designed to assist translators by decomposing the workflow into four main stages: speech bubble segmentation, optical character recognition (OCR), machine translation, and typesetting. Rather than replacing human translators, our system aims to reduce repetitive manual effort. A Qt-based application is developed to provide an interactive interface, enabling users to load manga pages and visualize translated outputs in a unified environment. All processing stages are executed automatically in the background, allowing translators to focus on translation refinement.

1 Introduction

Among recent forms of entertainment, digital comics have become increasingly important, especially for young audiences, due to their distinctive hand-drawn art styles and engaging storylines. Comics not only provide entertainment but also foster imagination, creativity, and visual literacy in readers. Their narratives are conveyed primarily through illustrations, making complex ideas and emotions more accessible to younger minds. The diversity of comic content is expanding as creators can easily publish online and reach a wide audience. Manga, a style of comic originating in Japan, has gained significant global popularity—not only for its unique artistic style but also for its rich storytelling and cultural insights. To make manga accessible to global audiences, translation plays a vital role.

However, the process is highly labor-intensive because translators not only interpret the text, but also edit comic pages using graphic tools, remove original dialogues, and replace them with translated content. While translation itself is already a demanding task, the additional manual work significantly increases both time and effort, leading to exhaustion, low efficiency, and quality reduction which may impact readers' experience. The manual translation workflow typically consists of several stages, as illustrated in Figure 1. First, translators read the manga to identify key terms and establish a style guide. They then produce translation drafts while continuously updating the glossary. The final stage is typesetting, which is usually handled by letterers in large translation teams, or by the translators themselves in smaller teams. Images are edited to replace the original text with the translated text. This repetitive procedure can be exhausting for translators, as manga series are typically released over multiple volumes.

With the advancement of technology, comic translation applications were born to support translators and improve workflow efficiency. They can reduce manual workload by automating tasks so that translators can focus on delivering high-quality translations. We propose an end-to-end application supporting professional translators by extracting speech bubbles and text automatically, along with assisted translation features. This work aims to bridge computer vision and natural language processing to perform automatic manga translation.

Applying deep learning models and computer vision techniques to manga translation has become a popular topic in academic research. However, practical applications for manga translation are limited and still present several drawbacks for professional translators. Manga Translator [1], Comic Translate [2], and Mantra Engine [3] are representative applications developed for manga translation, corresponding respectively to web, desktop, and mobile platforms.

All of these applications offer fast processing with real-time or near–real-time performance, making them suitable for personal use or beginners. Some systems also allow users to select different translation models or support multiple languages, font styles, and image formats. Nevertheless, they share a critical limitation: none of them provide an editing function for the translated text.

In addition, Comic Translate [2] requires users to obtain and configure their own API keys, which can be challenging for non-technical users. The mobile platform, Mantra Engine [3], produces visually unrefined text overlays that often exceed the boundaries of speech bubbles, negatively affecting readability and aesthetic quality.

Therefore, our motivation is to develop a platform specifically designed for professional manga translators that provides editing functionality at each stage of the manga translation pipeline. Rather than fully replacing human translators, the application aims to support and enhance manual translation workflows by integrating existing deep learning models and computer vision techniques.

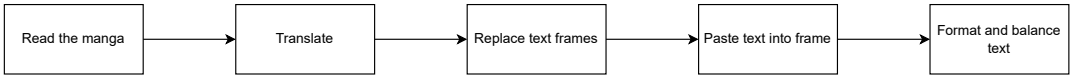


Figure 1: Manual manga translation workflow

2 Related Works

Manga translation is not a new research topic since there are many publications working on automatic manga translation. In 2022, Saikumar et al. [4] presented an automated system for translating manga and integrating a multi-stage process into a functional framework. Authors mentioned a common issue with Japanese while performing translation: Japanese lacks spaces between words and its word ordering in manga (from left to right vertically) are challenging to conventional OCR. Another paper by Hinami et al. [5] raised another difficulty in context understanding since a character can be divided up into multiple bubbles (narrations) and the order of speech bubbles.

2.1 Speech Bubble Segmentation

Speech balloon segmentation is a fundamental step in manga translation pipelines, as it localizes dialogue regions and defines the basic units for subsequent OCR and translation. Despite its importance, the number of dedicated studies on speech balloon segmentation remains relatively limited. Early in 2013, Rigaud et al. [6] detailed a speech balloon detection method using active contour framework. Edge-based image features and text localization were combined to guide the evolution of contours towards speech balloon boundaries. Two years later, Rigaud et al. [7] proposed an adaptive thresholding method to binarize grayscale images and extract connected components, identifying speech balloons based on their content topology and alignment. In 2019, Dubray and Laubrock [8] employed a deep convolutional network with a U-Net architecture and a pre-trained VGG-16 encoder to segment speech balloons in comics. Later, Melista et al. (2021) [9] combined Faster R-CNN for balloon bounding box detection with U-Net for pixel-level segmentation of speech balloons. Besides, post-processing for overlapping speech balloons has also been an interesting problem for computer vision researchers. In 2015, Liu et al. [10] proposed a rule-based approach that utilized region (background) extension and heuristic rules to identify and split overlapping speech balloons.

2.2 Optical Character Recognition (OCR)

Optical character recognition (OCR) plays a critical role in manga translation by converting text within segmented speech balloons into machine-readable form. Early in 2012, Ranjini and Sundaresan [11] employed connected component analysis and edge-based features to extract English text from comic images. A similar text detection task was later addressed by Aramaki et al. [12] in 2016, who combined connected-component-based and region-based classification. In the same year, Hirata et al. [13] employed an image operator learning method generating binary segmentation in pixel level. Machine learning algorithms were applied to learn image operators from training images with ground truths, where windows are considered as features. The choice for window value is important in this task, which can lead to large generalization error. Therefore, two-level operators were born to solve the problem by using moderate size windows for training and combining resulting images. Later in 2018, Chu et al. [14] proposed two approaches based on deep networks for text detection. The first approach utilized multiple CNNs for feature extraction, joined them and fed to a combination of a classification and a regression network. The other approach took region proposal, extracted features and classification/regression in a single deep network. While previous studies primarily focus on developing specific text detection or segmentation methods, Soykan et al. [15] provide a comprehensive benchmark and evaluation framework that offers a unified perspective on comics OCR and facilitates systematic comparison across different approaches.

2.3 Layout Analysis and Reading Order in Comics

Layout analysis and reading order estimation are fundamental to comic understanding. These topics have attracted considerable research attention, as evidenced by several publications in the literature. In 2012, Ngo et al. [16] emphasized how graphic elements such as panels and speech balloons affected to full text indexing, therefore, proposed a region-based and morphology-based method to extract panels in a comic page. Two years later, Pang et al. [17] presented a robust method for panel extraction in comics by first closing open panels and identifying background masks. The panel block is then recursively splitted the panel into sub-blocks from which panel shapes are recovered. In 2019, Nguyen et al. [18] had a different approach by considering panel extraction in image segmentation task. They adopted U-Net architecture to perform multiclass classification for each pixel, whether it belongs to the background, panels or borders. In 2022, Omori et al. [19] proposed algorithms to estimate the reading order of comic frames by sorting and finding rightmost or topmost frame. If it exists, the first frame will be chosen according to its position in the sorted list of frames. If there is no topmost or rightmost frame, the first frame in the sorted output will be considered as the first.

2.4 Machine Translation and Typesetting

Manga translation is an increasingly popular topic in applied research; however, there is a lack of academic studies focusing specifically on Japanese-to-English manga translation. Early in 1986, Nagao et al. [20] described the outline of Japanese to English machine translation system. In 2021, Zhou et al. (2021) [21] introduced NiuTrans, built on variants of Transformer [22], Transformer-DLCL (Dynamic Linear Combination of Layers) [23], ODE (Ordinary Differential Equation)-Transformer [24] and vice versa. Besides, back-translation, knowledge-distillation, post-ensemble and iteractive fine-tuning techniques were applied to improve model performance. Later in 2024, Kinugawa et al. [25] reported the findings of the WMT 2024 shared task on non-repetitive translation, which is particularly relevant to dialogue translation scenarios where avoiding repetitive expressions is crucial. In the same year, Kaino et al. [26] proposed two new approaches to capture contextual information in machine translation for manga: scene-based translation and machine translation incorporating bibliographic attributes: series, author, publisher, magazine and genre. In manga translation pipelines, translation is typically followed by the typesetting phase,

which encompasses both the reinsertion of translated text into speech balloons and the removal of original text to prepare clean background regions. Within this context, text removal is a crucial supporting step for typesetting. In 2020, Ko et al. [27] proposed an automated framework for text removal in comics that integrates deep learning with image processing techniques. The framework consists of two main stages: pixel-level text segmentation followed by text erasure using inpainting to restore the surrounding visual content.

3 Materials and Methods

3.1 Overall Pipeline

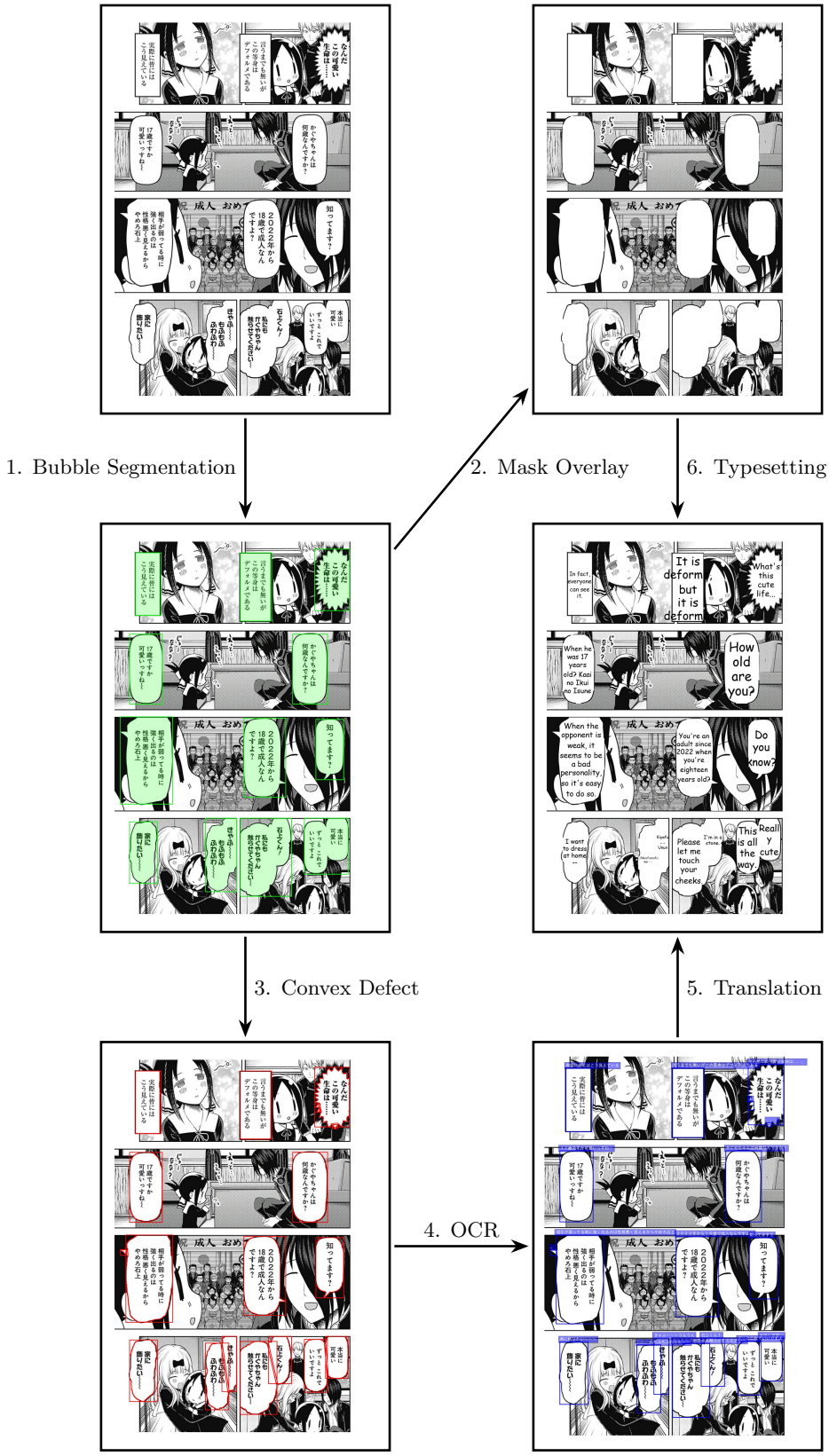


Figure 2: Pipeline Workflow

Figure 2 illustrates the overall pipeline of our manga translation system. First, speech bubbles in the input image are segmented using a YOLO-based model, producing instance-level bounding boxes and original segmentation masks. These masks are then refined using convex defect analysis to improve the geometric quality of bubble regions and to separate overlapping bubbles. Next, `manga-ocr` is applied to recognize Japanese text within the refined speech bubble regions. The recognized text is translated to English using `elan-mt-ja-en`. Finally, the translated text is rendered onto the original image based on the original segmentation masks, preserving the visual appearance of the speech bubbles in the final output.

3.2 Dataset

Manga109 is a dataset compiled by Aizawa Yamasaki Matsui Laboratory, Department of Information and Communication Engineering, the Graduate School of Information Science and Technology, the University of Tokyo [28, 29]. This dataset consists of 109 manga volumes with their dialog and segmentation annotations, intended for use in academic research.

The dataset contains 10,607 images, of which 9,916 include speech bubbles, with a total of 130,176 bubbles. Segmentation annotations are stored in in COCO-format JSON files. Bubble masks and bounding boxes are stored in Run-Length Encoding (RLE) and [x,y,width,height] formats, correspondingly. Since RLE is not suitable for model training, evaluating or result visualisation, it must be converted into polygon representations. In this work, only images containing annotated speech bubbles are used for model training and evaluation. The maximum number of speech bubbles in a single image is 44, and on average, each image contains approximately 13 bubbles, indicating a relatively high bubble density. The bubble distribution following COCO standard is illustrated in Figure 3.

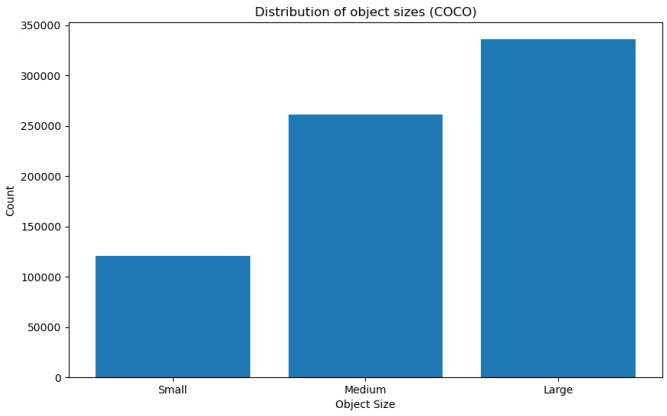


Figure 3: Speech bubble distribution in Manga109 dataset following COCO standard.

Figure 3 illustrates that speech bubbles considered as large objects make up 46.8% in total. Medium and small bubbles account for 36.4% and 16.8%, respectively. This distribution indicates a significant presence of large speech bubbles in the dataset, which may influence the performance of detection models, as larger objects are generally easier to detect than smaller ones.

3.3 Bubble Segmentation

3.3.1 Pipeline

The bubble segmentation pipeline systematically processes raw manga pages—whether single-page or two-page spreads—through three sequential stages to extract individual speech bubble regions suitable for OCR processing, as illustrated in Figure 4.

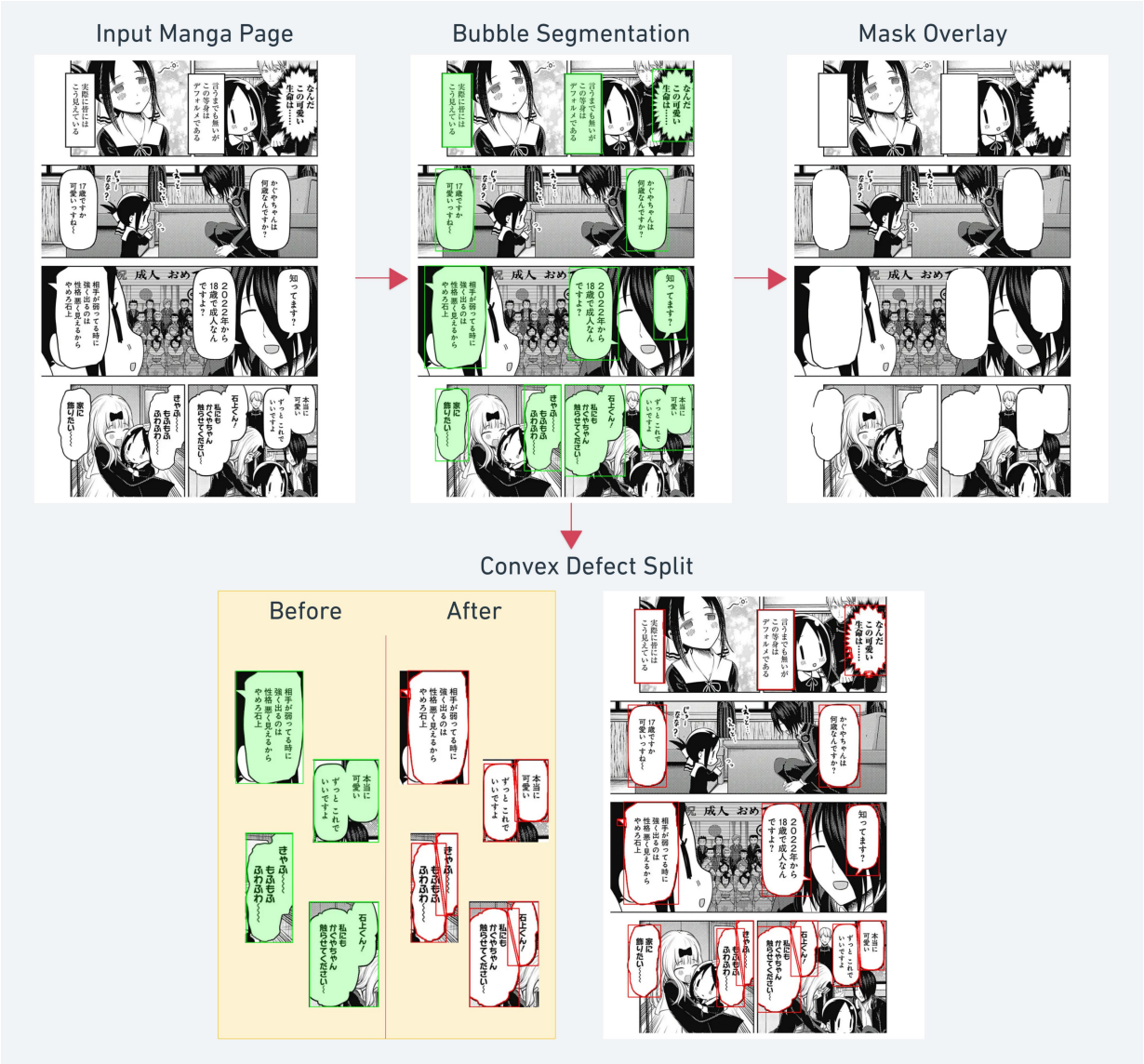


Figure 4: Bubble Segmentation Pipeline

The input manga page is processed through the YOLOv8 instance segmentation model, which generates pixel-precise segmentation masks for each detected speech bubble. These masks are visualized as green overlays in the figure, delineating the spatial extent of individual bubble regions across the page.

A critical step in the pipeline is overlaying the segmentation masks onto the original manga page to systematically erase all Japanese characters within the detected bubble regions. This mask-based whitening operation serves as an essential preparatory step for subsequent text replacement, ensuring clean canvas areas for the insertion of translated text.

While YOLO performs robust segmentation for isolated speech bubbles, it exhibits a fundamental limitation: when multiple bubbles are physically adjacent or overlapping, the model generates a single merged mask encompassing all connected regions. To address this deficiency, we implement a post-processing algorithm that analyzes convexity defects to identify merge boundaries and recursively splits connected bubble masks into discrete components. This splitting mechanism ensures that each speech bubble is processed independently during OCR, thereby maintaining text recognition accuracy.

3.3.2 Data Preparation

In this work, Manga109s, a subset of the Manga109 dataset [28, 29] containing 87 manga volumes, was utilised for model training. In total, the training data consisted of 7958 images with 97896 annotated speech bubbles. The remaining 22 manga titles were used for evaluating, including 1958 images with 29848 annotations.

Each JSON (JavaScript Object Notation) annotation file contains six category identifiers, including our target annotation with category ID equal to 5. To reduce data retrieval time, related information including speech balloon annotations and associated metadata were filtered and stored in a dictionary, where each manga title corresponds to one sub-dictionary.

The dataset was then split and grouped at the manga volume level for training and testing. Images were copied to the corresponding directories and annotations were written to text files for each image. The dataset structure was specified in a YAML configuration file for model training and validation.

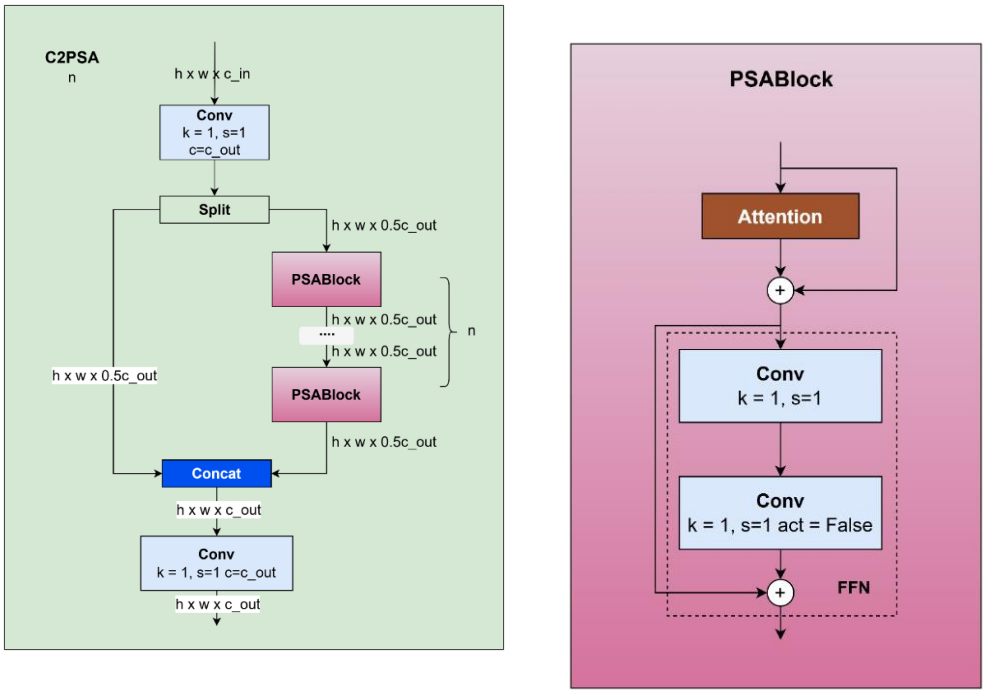
Finally, original speech bubble bounding boxes were defined in the [x, y, width, height] format whereas YOLO models expect bounding boxes in the [x_c, y_c, width, height] format. Therefore, bounding boxes were normalised for storage in text files. During evaluation, YOLO internally converted them into an appropriate representation required for metric computation.

3.3.3 YOLO Segmentation architecture

In this study, we employ the YOLO architecture adapted for instance segmentation tasks. Specifically, we investigate two versions: YOLOv8 [30] and YOLOv11 [31]. While both models share a fundamental Anchor-Free, One-Stage detection paradigm, YOLO11 introduces significant architectural refinements to improve feature extraction and processing speed. The segmentation architecture consists of three primary components: the Backbone, Neck, and Head.

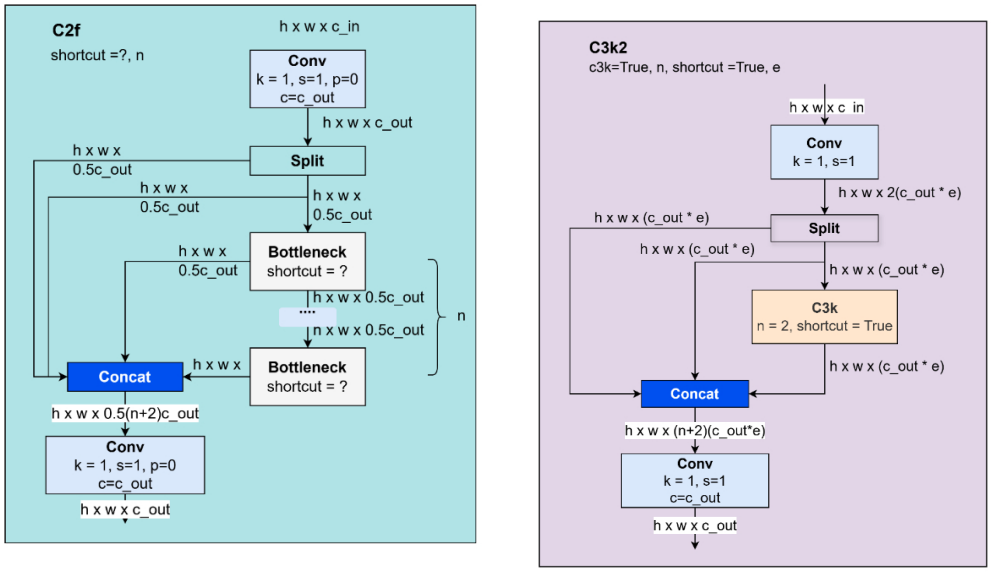
1. **Backbone:** The Backbone is responsible for extracting feature maps from the input manga pages. YOLOv8 utilises the CSPDarknet structure enhanced with C2f (Cross Stage Partial bottleneck with 2 convolutions) modules. The C2f module improves information flow by merging the concepts of C3 modules [32] and ELAN (Efficient Layer Aggregation Networks) [33].

   YOLOv11 replaces the C2f module with the C3k2 block (C3k with a customable kernel size). Furthermore and integrates the C2PSA (Convolutional Block with Parallel Spatial Attention) module at the end of the backbone [31]. This attention mechanism allows the model to focus more effectively on spatial features such as the contours of a bubble against complex manga backgrounds before passing data to the Neck. The architectures of the C2PSA module, the attention block in YOLOv11, the C2f module, and the C3k2 module are illustrated in Figures 5 [34] and 6 [34].



(a) C2PSA module (b) Attention block (YOLOv11)

Figure 5: Architecture of PSA-based modules: (a) C2PSA module; (b) Attention block used in YOLOv11 [34]



(a) C2f module (b) C3k2 module (YOLOv11)

Figure 6: Comparison of backbone modules: (a) C2f module; (b) C3k2 module adopted in YOLOv11 [34]

2. **Neck (Feature Fusion):** Both versions employ a PANet (Path Aggregation Network) [35] structure combined with an FPN (Feature Pyramid Network) [36]. This component aggregates features from different backbone stages. Since manga bubbles can range from tiny whispers to page-dominating shouts, this multi-scale fusion is critical for detecting instances of all sizes effectively.

3. **Segmentation Head for Mask Generation:** Unlike standard object detection, the output of segmentation head is pixel-wise mask, followed by Decoupled Head architecture [37], which separates the classification and regression tasks.

   Following the YOLACT (You Only Look At Coefficients) principle [38], the mask generation pipeline begins with Protonet, a dedicated branch that learns a fixed set of k prototype masks—generic bases that are shared across every image and never conditioned on any particular object. Concurrently, the detection head predicts, for each anchor, k scalar mask coefficients that encode the instance-specific contribution of every prototype. These coefficients are multiplied with their corresponding prototypes, the weighted maps are summed, a sigmoid activation converts the logits to per-pixel probabilities, and the resulting mask is finally cropped to the predicted bounding box to produce the assembled instance mask. The architecture of YOLACT is illustrated in Figure 7.

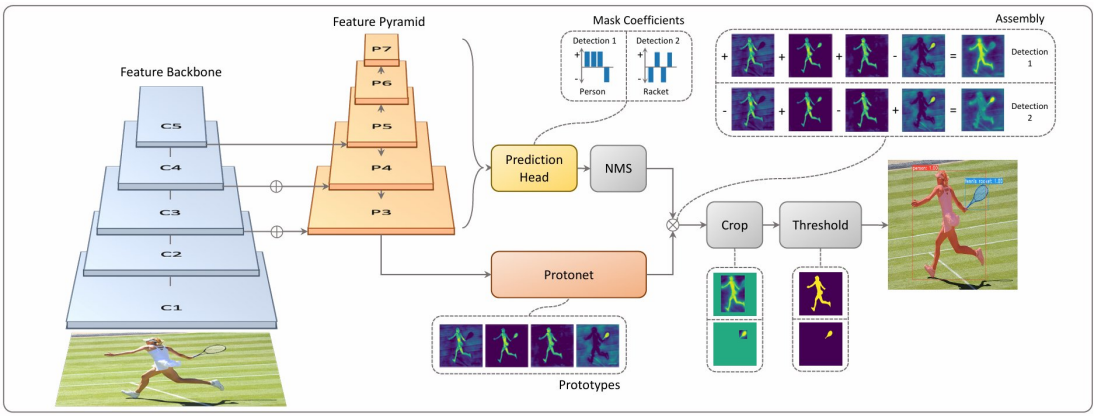


Figure 7: YOLACT architecture

3.3.4 Model Training

Four YOLO models—YOLOv8s, YOLOv8n, YOLOv11s, and YOLOv11n—were trained using the same hyper-parameters, as shown in Table 1. Apart from the defined hyper-parameters, all other training settings remained unchanged. In each epoch, images were processed in batches in parallel. Images were resized to the selected input resolution, as specified in Table 1 and then forwarded through the layers of the YOLO models. After each epoch, the validation loss was computed, and the model achieving a lower validation loss than the current best was saved as the best-performing

model. Model parameters were updated via back propagation and used in the next epoch.

Hyper-parameter	Value
Epochs	10
Image size	1280×1280
Batch size	4

Table 1: Training hyper-parameters for YOLO models

3.3.5 Post-processing for speech bubbles

Although YOLO gives a pixel-precise mask for every panel, it still returns one mask when multiple balloons touch or overlap. Feeding a merged region to an OCR engine makes the text layout denser and usually causes a noticeable drop in recognition accuracy.

To address this, we introduce a post-processing step that leverages convexity defects to detect and split merged speech bubbles. The key observation is that when two bubbles are connected, the resulting contour often exhibits at least two deep concave regions — known as convexity defects. These defects serve as reliable cues for where a cut should be made. The parameters for this algorithm are illustrated in Table 2.

Table 2: Post-processing Parameters for Bubble Splitting

Parameter	**Value**	**Rationale**
MIN_DEFECT_DEPTH	13 px	Minimum depth of a defect to be considered
MAX_ANGLE_DEG	170°	Maximum angle of a defect to be considered
MIN_DIST	20 px	Minimum distance between two defects to be considered
MIN_AREA	100 px^2	Minimum area of a connected component to be considered
MAX_RECURSION	2	Maximum recursion depth

Our algorithm proceeds as described in Algorithm 1 and 2: for each mask returned by YOLO we extract the largest external contour, compute its convex hull and identify convexity defects—points where the contour deviates remarkably from its convex envelope—retaining only those that are both deep and angularly valid to suppress noise, then merge nearby defects to avoid over-segmentation; subsequently we draw a straight line between the closest pair of defects if at least two valid ones remain, whereas if only a single valid defect exists we skip the cut entirely since one defect alone does not furnish sufficient geometric evidence for a reliable split, and the entire process is applied recursively to each resulting sub-mask up to a predefined depth to handle more complex multi-bubble clusters.

The splitting procedure operates in two tightly-coupled functions. `SplitConnectedBubbles` serves as the recursive driver: it invokes `AttemptSplitOnce` on the input mask and, if more than one sub-mask is returned, recursively processes each fragment until the maximum recursion depth is reached. `AttemptSplitOnce` encapsulates the core geometric logic. It first extracts the largest contour and computes its convex hull. Using OpenCV's `convexityDefects`, it identifies points where the contour bends inward significantly. Each defect is characterized by four values: the start point s, end point e, farthest point f, and depth d. We retain only defects whose depth exceeds `MIN_DEFECT_DEPTH` (13 pixels) and whose interior angle—computed via the dot product of vectors $\vec{v_1} = s - f$ and $\vec{v_2} = e - f$—is less than `MAX_ANGLE_DEG` (170°). This angular constraint filters out shallow, wide concavities that are unlikely to represent merge boundaries.

Algorithm 1 Split Merged Speech Bubbles Using Convexity Defects (Part 1)

Input: M (Binary mask of detected bubble), *depth* (current recursion depth)
Output: List of separated bubble masks
function SplitConnectedBubbles($M, depth$)
    if $depth \geq$ MAX_SPLIT_DEPTH **then**
        return $[M]$
    end if
    $sub\_masks \leftarrow$ AttemptSplitOnce(M)
    if $|sub\_masks| = 1$ **then**
        return sub_masks
    end if
    $result \leftarrow \emptyset$
    for all $sub \in sub\_masks$ **do**
        $result \leftarrow result \cup$ SplitConnectedBubbles($sub, depth + 1$)
    end for
    return $result$
end function

Algorithm 2 Split Merged Speech Bubbles Using Convexity Defects (Part 2)

Input: M (Binary mask of detected bubble)
Output: List of separated bubble masks
function AttemptSplitOnce(M)
    $contour \leftarrow$ LargestContour(M)
    if $contour = \emptyset$ **or** Area(contour) < 500 **then**
        return $[M]$
    end if
    $hull \leftarrow$ ConvexHull($contour$)
    $defects \leftarrow$ ConvexityDefects($contour, hull$)
    if $defects = \emptyset$ **then**
        return $[M]$
    end if
    $candidates \leftarrow \emptyset$
    for all $(s, e, f, d) \in defects$ **do**
        $depth\_val \leftarrow d/256.0$
        if $depth\_val >$ MIN_DEFECT_DEPTH **then**
            $\vec{v_1} \leftarrow s - f$, $\vec{v_2} \leftarrow e - f$
            $angle \leftarrow \arccos\left(\frac{\vec{v_1} \cdot \vec{v_2}}{|\vec{v_1}||\vec{v_2}|}\right)$
            if $angle <$ MAX_ANGLE_DEG **then**
                $candidates \leftarrow candidates \cup \{(f, depth\_val)\}$
            end if
        end if
    end for
    $candidates \leftarrow$ SortByDepth($candidates, descending$)
    $unique \leftarrow \emptyset$
    for all $c \in candidates$ **do**
        if $\forall u \in unique : \|c.point - u.point\| \geq$ MIN_DIST **then**
            $unique \leftarrow unique \cup \{c\}$
        end if
    end for
    $M_{cut} \leftarrow M$
    $cut\_happened \leftarrow false$
    if $|unique| \geq 2$ **then**
        $top4 \leftarrow$ first 4 elements of $unique$
        $(p_1, p_2) \leftarrow$ ClosestPair($top4$)
        DrawLine(M_{cut}, p_1, p_2, color=0, thickness=3)
        $cut\_happened \leftarrow true$
    end if
    if $cut\_happened$ **then**
        $contours_{new} \leftarrow$ FindContours(M_{cut})
        $masks_{new} \leftarrow \emptyset$
        for all $c \in contours_{new}$ **do**
            if Area(c) > 100 **then**
                $m \leftarrow$ CreateMaskFrom(c)
                $masks_{new} \leftarrow masks_{new} \cup \{m\}$
            end if
        end for
        if $|masks_{new}| > 1$ **then**
            return $masks_{new}$
        end if
    end if
    return $[M]$
end function

After collecting candidates, we cluster them by sorting in descending order of depth and discarding any candidate within `MIN_DIST` (20 pixels) of a previously retained one. If at least two unique

defects remain, we select the closest pair from the top four deepest candidates and draw a black line between them, effectively severing the mask. Only when the cut produces multiple connected components with area greater than 100 pixels do we accept the split; otherwise, the original mask is returned unchanged. The recursion limit of 2 prevents excessive fragmentation while still handling typical two- or three-bubble clusters observed in manga panels.

3.3.6 Post-processing for panel detection

Although YOLO provides instance masks for panels, it occasionally generates fragmented or overlapping detections for a single panel. To address this, we implement a merging strategy based on the Intersection-over-Minimum-Area (IoM) ratio. Unlike standard Intersection-over-Union (IoU), IoM is robust against cases where a fragmented detection is significantly smaller than (and contained within) the main panel prediction.
For any pair of detected panels with bounding boxes B_i and B_j, we calculate the overlap ratio R_{merge}:

$$R_{\text{merge}} = \frac{\text{area}(B_i \cap B_j)}{\min(\text{area}(B_i), \text{area}(B_j))} \tag{1}$$

We utilize a threshold of $\tau = 0.2$ for the merging decision. This value was *determined empirically* to aggressively merge fragmented predictions while preserving distinct but adjacent panels. If $R_{\text{merge}} > \tau$, the two panels are merged into a single instance: the new bounding box is the union of coordinates, and the new mask is the pixel-wise logical disjunction (OR) of the original masks. This process is repeated iteratively until no overlapping pairs satisfy the condition.

3.4 Layout Analysis and Bubble Ordering

Preserving the correct reading order is critical for the translation to be coherent. Japanese manga follows a specific Right-to-Left (RTL), Top-to-Bottom flow that differs from Western comics. We propose a hierarchical layout analysis pipeline that structures the unstructured bubble detections into a linear narrative sequence.

3.4.1 Double-page Processing

Standard manga processing pipelines often fail on double-page spreads (two facing pages scanned as a single image), as the layout analysis algorithms assume a single flow of panels. We address this by detecting spreads via aspect ratio analysis. If the image width dominates the height ($W/H > 1.0$), we trigger a specialized split-processing routine.

The image is split vertically at $x_{\text{mid}} = W/2$ into two sub-images: the Right Page (I_R) and the Left Page (I_L). In adherence to the Japanese reading direction, I_R represents the earlier page in the narrative sequence and is processed first.

After independent segmentation inference on I_R and I_L, the results must be mapped back to the original full-page coordinate system.

- **Bounding Box Reprojection:**
  - For the Left Page (I_L), coordinates remain unchanged: $B_{\text{global}} = B_{\text{local}}$.
  - For the Right Page (I_R), we apply a horizontal offset to map the "local" coordinates back to the right half of the canvas:

$$x_{\text{global}} = x_{\text{local}} + x_{\text{mid}} \tag{2}$$

- **Mask Reconstruction:** The instance masks returned by the model correspond to the dimensions of the sub-images. To reconstruct the full-page masks, we initialize a zero-filled canvas of size $W \times H$ for each detection. The local mask is then embedded into the canvas at the corresponding ROI (Region of Interest):
  - $M_{\text{global}}[y, 0 : x_{\text{mid}}] = M_{\text{local}}$ (Left Page)
  - $M_{\text{global}}[y, x_{\text{mid}} : W] = M_{\text{local}}$ (Right Page)

Finally, the lists are concatenated in the order $[L_{\text{right}}, L_{\text{left}}]$ to ensure the reading order flows correctly from the first page (right) to the second (left).

3.4.2 Hierarchical Ordering

The ordering process operates on two levels: Panel-level and Bubble-level.

- **Panel Ordering:** We divide the page vertically into logical rows. Panels are assigned to rows based on their vertical center coordinates. Rows are processed top-to-bottom, and within each row, panels are sorted right-to-left.
- **Bubble Assignment:** Each speech bubble is assigned to a panel if its centroid lies within the panel's bounding box. Bubbles that fall outside all panels are marked as "unassigned" and appended to the end of the sequence.
- **Intra-panel Ordering:** Inside each panel (or for the unassigned group), bubbles often do not follow a strict grid. We employ an adaptive row grouping algorithm to sort them:

Algorithm 3 Adaptive Bubble Ordering

```
procedure AdaptiveBubbleOrdering(B)
    Input: List of bubbles B = {b_1, ..., b_n}
    Output: Ordered list L
    Sort B by vertical center y (ascending)
    rows ← ∅
    current_row ← {B[1]}
    τ_gap ← 0.5 × average_height(B)
    for i ← 2 to n do
        if (B[i].y − B[i − 1].y) > τ_gap then
            rows ← rows ∪ {current_row}
            current_row ← {B[i]}
        else
            current_row ← current_row ∪ {B[i]}
        end if
    end for
    rows ← rows ∪ {current_row}
    L ← ∅
    for row ∈ rows do
        Sort row by horizontal center x (descending)        ▷ Right-to-Left
        L ← L ∪ row
    end for
    return L
end procedure
```

3.4.3 Handling Cross-Page Elements

A specific challenge arises when a panel or a speech bubble spans across the centre spine (the split line). In our current pipeline, these elements are effectively split into two separate instances:

- **Cross-Page Panels:** A single large panel spanning both pages is detected as two separate panels: one on the Right Page (P_R) and one on the Left Page (P_L). Since our ordering logic strictly sequences the Right Page before the Left Page, the narrative flow remains coherent: the reader consumes the content of P_R first, then moves to P_L.
- **Cross-Page Bubbles:** Similarly, a bubble lying on the cut line may be detected as two distinct bubbles. While this generates two separate translation blocks, the temporal ordering is preserved ($\text{Bubble}_R \rightarrow \text{Bubble}_L$).

3.5 Optical Character Recognition (OCR)

To translate the visual content of speech bubbles into machine-readable text, we integrate the **MangaOCR** framework. While general-purpose OCR engines (e.g., Tesseract [39]) struggle with the non-linear layout and unique typography of Japanese manga, MangaOCR is specifically architected to handle vertical text, **furigana** (pronunciation guides), and mixed writing systems.

3.5.1 Pipeline

The OCR pipeline processes segmented speech bubble regions through a systematic sequence of extraction, recognition, and refinement operations to convert visual Japanese text into machine-readable characters, as illustrated in Figure 8.
The pipeline receives as input the full manga page with text regions detected and isolated following the convex defect splitting procedure. For each detected speech bubble, we extract the corresponding bounding box coordinates and crop the region from the original image, thereby isolating individual text-bearing areas for independent processing.

Each cropped bubble region is passed through the MangaOCR model, which performs character-level recognition of vertical Japanese text. The model leverages its vision encoder-decoder architecture to generate sequential character predictions, producing raw text output for each bubble region.

The raw OCR output undergoes three sequential post-processing techniques to enhance text quality and correct common recognition errors. These refinement operations address character-level mistakes, normalize spacing and punctuation, and apply linguistic corrections to improve the overall accuracy of the recognized text before it is passed to the translation module.

3.5.2 Model architecture

We utilize the official pre-trained weights of the `manga-ocr-base` model. Training a robust manga OCR model from scratch requires generating millions of synthetic image-text pairs and significant computational resources to converge. By leveraging the pre-trained model, which has already learned high-level feature representations from extensive synthetic datasets (based on CC-100 corpus and Manga109), we achieve state-of-the-art performance without the overhead of training. The architecture of the Manga-OCR model is illustrated in Figure 9.
The model follows a **Vision Encoder-Decoder** architecture with approximately 115 million parameters:

1. **Vision Encoder (ViT):** The backbone is a Data-efficient Image Transformer (DeiT-Tiny) [40] with 12 layers. It processes the input manga bubble image (224×224 pixels) by first splitting it into 196 non-overlapping patches of 16×16 pixels each. Each patch is converted into a 768-dimensional embedding vector and augmented with positional information to preserve spatial relationships. The patch sequence then passes through 12 Transformer encoder layers, where Multi-Head Self-Attention (MSA) mechanisms capture both local character

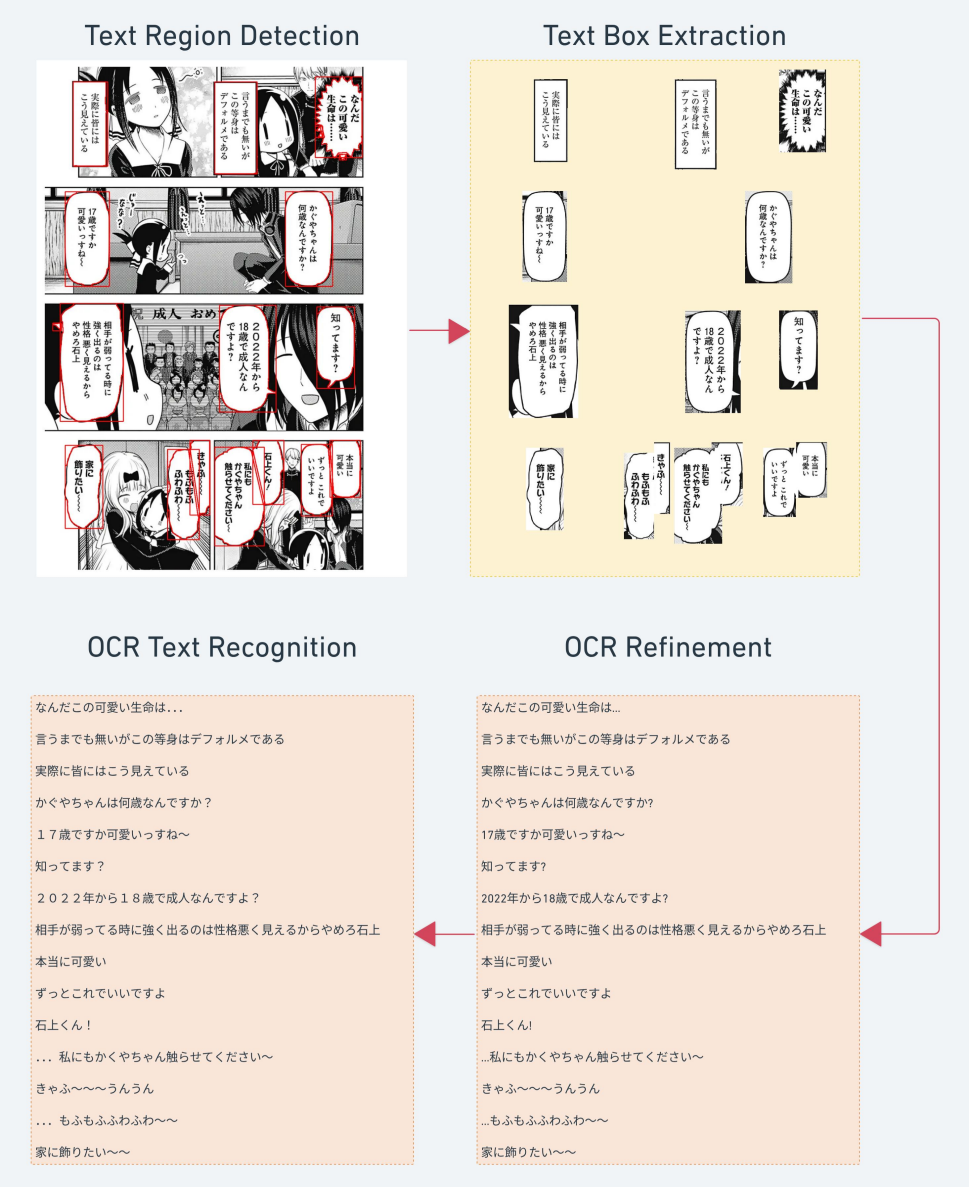


Figure 8: OCR Pipeline

strokes and global text layout patterns. The output is a set of visual features H_{enc} that encode the entire image content.

2. **Text Decoder (BERT):** The decoder is a Japanese character-level BERT model [41] adapted for autoregressive text generation. It generates characters one at a time by processing previously generated text (context tokens) through masked self-attention layers, ensuring each prediction only depends on prior characters. The decoder's key innovation is the **Cross-Attention** mechanism, which connects visual features H_{enc} from the encoder to the text generation process in H_{dec}. This mechanism allows the model to focus on relevant

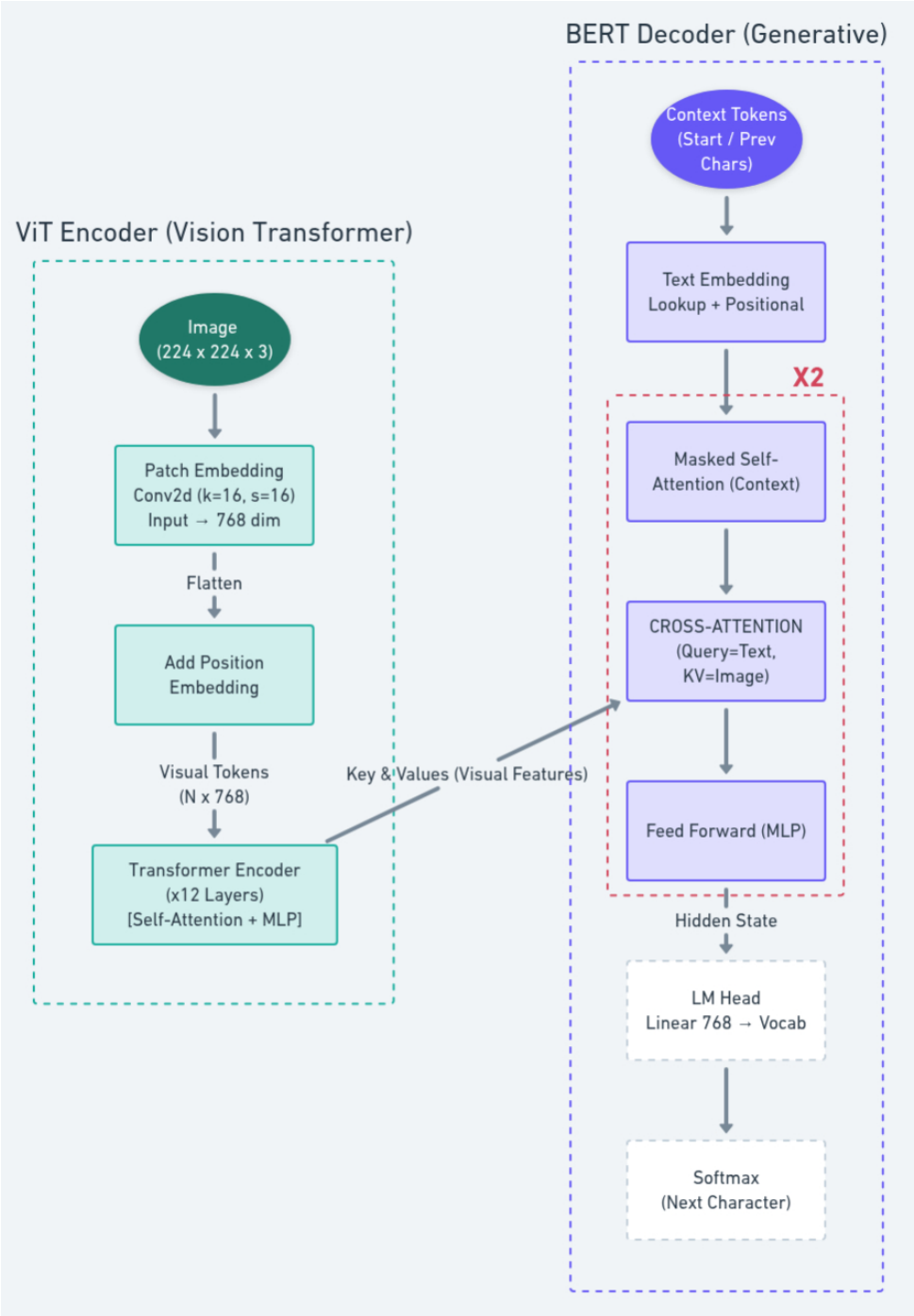


Figure 9: Manga-OCR Model Architecture

parts of the image—such as specific character strokes—when predicting each character. The attention operation is computed as:

$$Q = H_{\text{dec}} W_Q, \quad K = H_{\text{enc}} W_K, \quad V = H_{\text{enc}} W_V \tag{3}$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{4}$$

After cross-attention integrates visual information with textual context, the final hidden state

passes through a linear projection (LM Head) that maps to the vocabulary of 4,718 Japanese characters, followed by softmax to produce probability distributions over the next character.

Where Q (Query) denotes the textual context from decoder states H_{dec}; K (Key) and V (Value) denote visual features from encoder states H_{enc}; W_Q, W_K, and W_V are learnable projection matrices; and d_k is the key dimensionality used for gradient stabilization.

3.5.3 Inference algorithm

The inference pipeline transforms the raw crops from the segmentation module into a coherent dialogue transcript.
We employ a centroid-based algorithm to map text annotations to bubble regions. Let b be a detected bubble mask and t be a text bounding box with center $C(x, y)$. The association function $f(t, b)$ is defined as:

$$f(t,b) = \begin{cases} 1 & \text{if } C(x,y) \in \text{Area}(b), \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

For each sorted crop, we generate text using **Beam Search** (width $k = 4$). We apply a length penalty $alpha = 2.0$ to the scoring function to encourage the generation of complete sentences, ensuring that longer dialogue lines are not truncated prematurely.

3.5.4 Post-processing for OCR results

The raw output generated by the OCR model often contains artifacts inherent to stochastic decoding, such as irregular whitespace and inconsistent character encodings. To ensure high-quality input for the subsequent translation module, we implemented a deterministic cleaning pipeline utilizing regular expressions and Unicode normalization.
The process applies the following transformations sequentially:

1. **Unicode Normalization (NFKC):** Standardizes character widths by converting full-width alphanumeric characters (e.g., １ Ａ) into their canonical half-width counterparts (“1”, “A”).
2. **Whitespace Removal:** Eliminates phantom spaces inserted between characters, as Japanese writing does not rely on whitespace for word delimitation.
3. **Punctuation Standardization:** Collapses sequences of multiple periods into a single ellipsis (...) and normalizes tildes (~) to the Japanese wave dash (U+301C) to maintain typographic consistency.

The impact of this pipeline is demonstrated in Table 3.

Table 3: Comparison of Raw OCR outputs against Post-processed text and Ground Truth.

Ground Truth	Raw OCR Output	Post-processed Text
爆発まで残り 04:30:21	爆発まで残り、 Ｄ４：３０：２１	爆発まで残り、D4:30:21
なっななな...何やってんの!?	なっななな...何やってんの！？	なっななな...何やってんの!?
うへ~~!? なんじゃ	うへ～～!? なんじゃ	うへ~~!? なんじゃ
ファンクラブ規約No.13	ファンクラブ規約 Ｎｏ．１３	ファンクラブ規約No.13

3.6 Translation

3.6.1 Sentence-level translation

For single sentence translation, we utilised the pretrained 'elan-mt-ja-en' model series provided by Mitsua, which is built upon the MarianMT framework [42]. The underlying architecture of the model family is a standard Transformer encoder-decoder network [22].

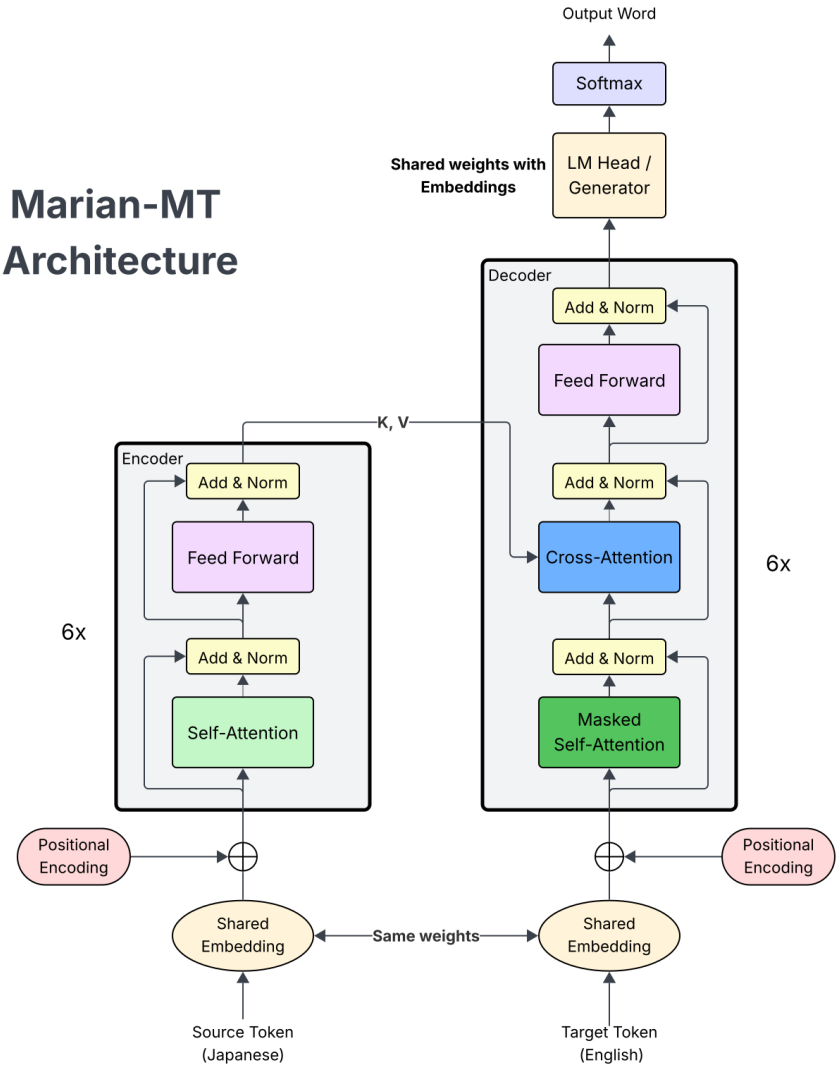


Figure 10: MarianMT Architecture

Model Architecture The MarianMT architecture follows the original Transformer design, consisting of an encoder that processes the source Japanese text and a decoder that generates the target English translation autoregressively. The encoder employs multi-head self-attention layers

to capture contextual relationships within the input sequence, while the decoder uses both self-attention and cross-attention mechanisms to attend to relevant encoder representations during generation.
The model utilizes SentencePiece tokenization [43], a subword segmentation approach that handles the morphologically rich nature of Japanese text by decomposing words into smaller units. This enables the model to handle rare words and out-of-vocabulary tokens effectively while maintaining a manageable vocabulary size.

Model variants We integrate three specific variants of the model to balance translation quality and computational efficiency, as shown in Table 4.

Model Name	Parameters	Description
Mitsua/elan-mt-bt-ja-en	61M	Back-translation augmented variant with improved handling of colloquial expressions
Mitsua/elan-mt-base-ja-en	61M	Base model trained on parallel corpora
Mitsua/elan-mt-tiny-ja-en	15M	Lightweight variant optimized for inference speed

Table 4: Variants of the Mitsua/elan-mt model family

The bt (back-translation) variant employs data augmentation through synthetic parallel data generation, where monolingual English text is translated back to Japanese to create additional training pairs.
The tiny variant reduces the model depth and hidden dimensions, achieving approximately 4× parameter reduction while maintaining reasonable translation quality. This variant is suitable for resource-constrained environments or applications requiring real-time inference.

3.6.2 Context-aware translation

Language constructs in Japanese are highly context-dependent, often omitting subjects or objects (pro-drop) known from history. While traditional NMT architectures can benefit from context, they typically require rigid input formats [5] (e.g., simple concatenation) and struggle to incorporate metadata such as the identity of the speaker, visual scene tags or page descriptions.
To address this, we approach translation not as a sequence to sequence task, but a instruction following task with context. We have chosen the **Qwen-2.5** instruction fine-tuned model family as the baseline for context-grounded translation.

Context schema We structured the input of the model as a "Chat" interaction. Unlike standard concatenation methods as in traditional NMT, we encapsulate the context in a structured json object. The schema consist of a **page_description** field, which contains the descriptions or overall context of the conversations being referenced, and a **target_bubble** field that contains the target bubble, which includes the speaker identity (either a name or an associated id) and the text contained inside the bubble. Additionally, the **prev_bubbles** and **next_bubble** fields contains a list of bubbles preceding and following the target bubble, respectivel. The length of these lists is the context window size, which can be controlled. Each bubble in these lists also contains the speaker identity and content like the target bubble.
Since extracting the **page_description** or speaker identity of each bubble is not always guaranteed, they can be set to "unknown", which notify the model that there is no context regarding those fields.

```
# INPUT

{
```

```
    "page_description": "meeting",
    "target_bubble": {
        "speaker": "アル ジョンソンさん",
        "text": "そうですね、A社の事はソフトウェアなのに、彼は造の仕事をしていましたよね。"
    },
    "prev_bubbles": [
        {
            "speaker": "ボブ クックさん",
            "text": "私もそう思ったのですが、ちょっと彼についてになることがあります。"
        },
        {
            "speaker": "アル ジョンソンさん",
            "text": "部のことですか? "
        },
        {
            "speaker": "unknown",
            "text": "なんか彼の去のはA社のとはないがするんですけどね。"
        }
    ],
    "next_bubbles": [
        {
            "speaker": "ボブ クックさん",
            "text": "そうですよね。"
        },
        {
            "speaker": "アル ジョンソンさん",
            "text": "だけど、部になるということは、客のための口になるということですよね。"
        },
        {
            "speaker": "ボブ クックさん",
            "text": "そうですよね、でも界に精通していることが大切だと思いませんか? "
        }
    ]
}

# EXPECTED OUTPUT

Well, Company A's business is related to software and he worked in the manufacturing industry."
```

Importantly, this schema remains compatible with sentence-to-sentence translation: by setting **page_description** and speaker to "unknown" and leaving **prev_bubbles** and **next_bubbles** as empty lists, the model receives only the isolated target sentence, effectively replicating standard NMT behavior within the same framework.

```
# INPUT

{
    "page_description": "unknown",
    "target_bubble": {
```

```
        "speaker": "unknown",
        "text": "そうですね、A社の事はソフトウェアなのに、彼は造の仕事をしていましたよね。"
    },
    "prev_bubbles": [],
    "next_bubbles": []
}

# EXPECTED OUTPUT

Well, Company A's business is related to software and he worked in the manufacturing industry."
```

Chat template formatting To ensure with the instruction-tuned Qwen2.5 base model, all training samples and inference inputs are formatted using the ChatML template. This template structures conversations with special tokens that mark the beginning and end of each message along with role identifiers.

Template Structure

```
<|im_start|>system
{system_message}<|im_end|>
<|im_start|>user
{user_message}<|im_end|>
<|im_start|>assistant
{assistant_response}<|im_end|>
```

During training and inference, the tokenizer will automatically convert the JSON template into the ChatML template.

Training dataset construction To train the model to utilise context robustly without losing its general translation capabilities, we constructed a unified instruction-tuning dataset by merging two distinct Japanese to English dataset sources.

Sentence to Sentence Sources

We combined large-scale sentence-level corpora including **JESC** which contains approximately 2.8 million Japanese-English sentence pairs after filtering. Being a crawled subtitle dataset, JESC is inherently noisier and may contain loose localizations or alignment errors typical of crowdsourced data. However, its inclusion is critical as it provides the vast volume of casual, colloquial speech patterns ($\approx$ 2.8M pairs) essential for manga translation, which formal datasets lack.
To counterbalance the noise from **JESC**, we incorporated **Flores+**, **NTREX**, and **Opus-100** (train split) as high-quality anchors. **Flores+** and **NTREX** are gold-standard benchmarks professionally translated to evaluate machine translation systems, ensuring high grammatical correctness and semantic fidelity. **Opus-100**, while larger and more diverse, serves as a strong general-domain baseline. Integrating these datasets prevents the model from overfitting to the loose, fragmented style of subtitles, grounding it in proper linguistic structures.

Context Rich Source

We integrated the **Business Scene Dialogue (BSD)** dataset [44] on the train split, which provides rich scene-tagged and speaker-identified dialogues in the 20000 Japanese-English sentence

pairs grouped into 670 distinct scenes. For these samples, we populated the JSON schema with large context window ($N = 5$ prior/next bubbles), explicit speaker names, and scene descriptions. The final training set mixes these two distributions. This hybrid approach teaches the model a dual capability: utilizing rich metadata when available (from the **BSD** samples) while falling back to robust sentence-level translation when no metadata is provided (from the General-Domain Sentence to Sentence sources).

Training configuration We fine-tune the **Qwen2.5-1.5B-Instruct** on the constructed dataset using LoRA (Low Rank Adaptation). During training with LoRA, we freezes the pre-trained model's weights and inject trainable rank decomposition matrices into one or several into specified layers of the model. This technique can significantly reduce the number of trainable parameter and VRAM requirements while being competitive to full supervised fine-tuning when trained on our dataset ($\approx$ 3.3M samples).
The training details of our model can be found in Appendix (7).

3.7 Typesetting

Typesetting is the final step in the manga translation pipeline. It takes the output of the post-processor and uses it to typeset the text in the panel. The typesetter is a simple function that takes a mask and a list of bounding boxes as input and returns a list of typesetted text. The typesetter uses the bounding boxes to determine the layout of the text and the mask to determine the content of the text.

Component	**Purpose**
Whitening	Erasing the original Japanese characters by filling the eroded mask region with white pixels.
Text Rendering	Rendering the text in the bubble mask.
Collision Detection	Ensuring that the rendered text does not collide with the bubble boundary.
Typesetting	Typesetting the text in the bubble mask.

Table 5: Typesetting Module Components

The typesetting pipeline operates in two sequential phases. In the whitening phase illustrated in Algorithm 4, we iterate over each bubble that contains translated text and erase the original Japanese characters by filling the eroded mask region with white pixels. Erosion with a 6×6 kernel prevents the fill from bleeding into the bubble's border, preserving the visual integrity of the speech balloon outline.

Algorithm 4 Render Translated Bubbles

function RENDER(I_{original}, B)
    $I_{\text{final}} \leftarrow \text{copy}(I_{\text{original}})$
    $K_{\text{erosion}} \leftarrow \text{ones}(6 \times 6)$
    Phase 1: Whitening (Inpainting) Step
    for all $b \in B$ **do**
        if b.translated_text is empty **then**
            continue
        end if
        $M \leftarrow b$.original_mask or b.mask
        $M_{\text{eroded}} \leftarrow \text{Erode}(M, K_{\text{erosion}})$
        $C \leftarrow \text{FindContours}(M_{\text{eroded}})$
        $\text{FillContours}(I_{\text{final}}, C, \text{white})$
    end for
    Phase 2: Text Rendering Step
    $I_{\text{pil}} \leftarrow \text{ConvertToPIL}(I_{\text{final}})$
    $D \leftarrow \text{CreateDrawContext}(I_{\text{pil}})$
    for all $b \in B$ **do**
        if b.translated_text is not empty **then**
            $\text{FitText}(D, b.\text{translated\_text}, b.\text{mask})$
        end if
    end for
    return $\text{ConvertToNumpy}(I_{\text{pil}})$
end function

The text rendering phase employs an adaptive font-sizing strategy. Starting from the largest feasible font size (the minimum of the bounding box's height and width), we progressively reduce the size in decrements of 2 until the wrapped text fits entirely within the bubble mask. The SmartWrapText function in Algorithm 5 handles word-level wrapping first; if a single word exceeds the available width, it falls back to character-level breaking to ensure no text overflows.

Algorithm 5 Smart Word Wrapping

function SmartWrapText($D, T, F, W_{\max}$)
  $words \leftarrow$ Split(T)
  $L \leftarrow []$ ▷ List of wrapped lines
  $current\_line \leftarrow []$
  for all $w \in words$ **do**
    $test\_line \leftarrow$ Join($current\_line + [w]$, space)
    $width \leftarrow$ MeasureTextWidth($D, test\_line, F$)
    if $width \leq W_{\max}$ **then**
      Append w to $current\_line$
    else
      if $current\_line \neq []$ **then**
        Append Join($current\_line$, space) to L
        $current\_line \leftarrow [w]$
      else
Force character-level breaking
        $temp \leftarrow$ ""
        for all $c \in w$ **do**
          if MeasureTextWidth($D, temp + c, F$) $\leq W_{\max}$ **then**
            $temp \leftarrow temp + c$
          else
            Append $temp$ to L
            $temp \leftarrow c$
          end if
        end for
        $current\_line \leftarrow [temp]$
      end if
    end if
  end for
  if $current\_line \neq []$ **then**
    Append Join($current\_line$, space) to L
  end if
  return L
end function

To guarantee that rendered text does not collide with the bubble boundary, we perform pixel-level collision detection described in Algorithm 6 and 7: the text block is temporarily drawn onto a blank canvas, and a bitwise and operation checks whether any text pixels overlap with the inverted bubble mask (i.e., the "wall" region outside the bubble). If collision is detected, we reduce the font size and retry. Should no valid size be found above the minimum threshold of 12 pixels, the algorithm renders the text at the minimum size regardless—ensuring the translation is always visible, albeit potentially clipped.

Algorithm 6 Fit Text in Mask with Collision Detection

function FitTextInMask(D, T, M)
  $C \leftarrow \text{FindContours}(M)$
  if $C = \emptyset$ **then**
    return
  end if
  $cnt \leftarrow \text{LargestContourByArea}(C)$
  $(x, y, w, h) \leftarrow \text{BoundingRect}(cnt)$
  $(c_x, c_y) \leftarrow \text{ComputeCentroid}(cnt)$
  $M_{\text{crop}} \leftarrow M[y : y + h, x : x + w]$
Attempt 1: Pixel-Perfect Fit with Binary Search
  $s_{\text{font}} \leftarrow \min(h, w)$
  $s_{\min} \leftarrow 12$
  $font_{\text{best}} \leftarrow \text{null}$
  while $s_{\text{font}} \geq s_{\min}$ **do**
    $F \leftarrow \text{LoadFont}(s_{\text{font}})$
    $L \leftarrow \text{SmartWrapText}(D, T, F, 0.95 \times w)$
    $H_{\text{text}} \leftarrow \sum_{l \in L} \text{TextHeight}(l, F)$
    if $H_{\text{text}} > h$ **then**
      $s_{\text{font}} \leftarrow s_{\text{font}} - 2$
      continue
    end if
Collision Detection via Mask Rendering
    $Canvas \leftarrow \text{CreateCanvas}(w, h, \text{black})$
    $y_{\text{curr}} \leftarrow (c_y - y) - H_{\text{text}}/2$
    for all $l \in L$ **do**
      $\text{RenderText}(Canvas, l, (c_x - x - w_l/2, y_{\text{curr}}), F, \text{white})$
      $y_{\text{curr}} \leftarrow y_{\text{curr}} + \text{TextHeight}(l, F)$
    end for
    if not $\text{CheckMaskCollision}(Canvas, M_{\text{crop}})$ **then**
      $font_{\text{best}} \leftarrow F$
      $lines_{\text{best}} \leftarrow L$
      $y_{\text{start}} \leftarrow c_y - H_{\text{text}}/2$
      break
    else
      $s_{\text{font}} \leftarrow s_{\text{font}} - 2$
    end if
  end while
Attempt 2: Fallback - Force Render at Minimum Size
  if $font_{\text{best}} = \text{null}$ **then**
    $font_{\text{best}} \leftarrow \text{LoadFont}(s_{\min})$
    $lines_{\text{best}} \leftarrow \text{SmartWrapText}(D, T, font_{\text{best}}, w)$
    $H_{\text{text}} \leftarrow \sum_{l \in lines_{\text{best}}} \text{TextHeight}(l, font_{\text{best}})$
    $y_{\text{start}} \leftarrow c_y - H_{\text{text}}/2$
  end if
Final Rendering
  $y_{\text{curr}} \leftarrow y_{\text{start}}$
  for all $l \in lines_{\text{best}}$ **do**
    $w_l \leftarrow \text{TextWidth}(l, font_{\text{best}})$
    $\text{DrawText}(D, l, (c_x - w_l/2, y_{\text{curr}}), font_{\text{best}}, \text{black})$
    $y_{\text{curr}} \leftarrow y_{\text{curr}} + \text{TextHeight}(l, font_{\text{best}})$
  end for
end function

Algorithm 7 Mask Collision Detection

function CHECKMASKCOLLISION($M_{\text{text}}, M_{\text{bubble}}$)
    $M_{\text{wall}} \leftarrow \text{BitwiseNot}(M_{\text{bubble}})$ ▷ White = Wall boundary
    $M_{\text{overlap}} \leftarrow \text{BitwiseAnd}(M_{\text{text}}, M_{\text{wall}})$
    return $\text{CountNonZero}(M_{\text{overlap}}) > 0$
end function

Finally, the text is drawn centered on the bubble's centroid (c_x, c_y), yielding a natural, balanced appearance consistent with professional manga typesetting conventions.

3.8 Evaluation metrics

3.8.1 Segmentation

As the proposed approach addresses an instance segmentation task, all four YOLO models are evaluated using the standard COCO evaluation protocol, reporting Precision, Recall, mAP50, and $mAP@[0.5 : 0.95]$ for both bounding box detection and mask segmentation. Precision and Recall are defined as:

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

Where TP, FP, and FN denote True Positives, False Positives, and False Negatives, respectively. A prediction is considered a true positive if its IoU with the ground-truth instance exceeds a predefined threshold; otherwise, it is counted as a false positive. Ground-truth bubbles that are not matched by any prediction are treated as false negatives.

Average Precision (AP) is computed as the area under the precision–recall curve. mAP50 denotes the mean AP at an Intersection over Union (IoU) threshold of 0.50, while $mAP@[0.5 : 0.95]$ averages AP over IoU thresholds from 0.50 to 0.95 with a step size of 0.05.

3.8.2 OCR metrics

For OCR evaluation, we use Character Error Rate (CER) and Word Error Rate (WER) as the primary metrics.

Character Error Rate (CER)
CER is derived as the Levenshtein distance from the predicted text and the reference text divided by total number of characters in the reference text, the metric is formulated as follow:

$$CER = \frac{S + D + I}{N} \tag{8}$$

Where S (Substitution) stands for the number of characters incorrectly recognized as different characters, D (Deletion) is the number of characters present in the reference but missing in the prediction, I (Insertion) is the number of extra characters present in the prediction but not in the reference and N defines the **total number of characters** in the reference text.

Word Error Rate (WER)

WER is widely used to evaluate the accuracy of transcription systems. It is derived as the Levenshtein distance from the predicted text and the reference text divided by total number of words in the reference text, the metric is formulated as follow:

$$WER = \frac{S + D + I}{N} \tag{9}$$

Where S (Substitution) is the number of words incorrectly recognized as different words, D (Deletion) is the number of words present in the reference but missing in the prediction, I (Insertion) stands for the number of extra words present in the prediction but not in the reference and N defines the **total number of words** in the reference text.

3.8.3 Translation metrics

For Japanese-to-English translation evaluation, we use COMETScore [45] and BERTScore [46] as the primary metrics, with chrF [47] and $chrF++$ [48] as additional metrics for reference.

COMETScore
COMET (Crosslingual Optimized Metric for Evaluation of Translation) is a neural-based metric that leverages pre-trained multilingual language models to evaluate translation quality. Unlike traditional n-gram-based metrics, COMET learns to predict human judgments by fine-tuning on human evaluation datasets. We use COMET-22 (wmt22-comet-da), the Direct Assessment model from WMT 2022.

$$COMET = f_\theta(src, hyp, ref) \tag{10}$$

Where f_θ defines a neural model (typically based on XLM-RoBERTa) that encodes the inputs, src is the source text, hyp is the hypothesis translation and ref is the reference translation.
COMET produces a quality score and has shown strong correlation with human judgments, being particularly effective for evaluating semantic adequacy.

BERTScore
BERTScore leverages pre-trained contextual embeddings from BERT to compute similarity between candidate and reference translations. Instead of exact n-gram matching, it computes semantic similarity at the token level. We use the DeBERTa-xlarge-mnli (microsoft/deberta-xlarge-mnli) backbone, which is the gold standard recommended model for BERTScore.

$$BERTScore_F = \frac{2 \cdot P_{BERT} \cdot R_{BERT}}{P_{BERT} + R_{BERT}} \tag{11}$$

Where P_{BERT} is the precision (matching tokens in candidate to reference) and R_{BERT} is the recall (matching tokens in reference to candidate).
Each token is represented by its contextual embedding, and similarity is computed using cosine similarity.

Character n-gram F-score (chrF and chrF++)
The Character n-gram F-score (chrF) is an evaluation metric that assesses the quality of machine translation by calculating the F-score of character n-gram overlaps between the hypothesis and the reference. Unlike word-level metrics, chrF works at the character level, making it particularly robust for morphologically rich languages and less sensitive to tokenization errors.

$$chrF = \frac{(1+\beta^2) \cdot chrP \cdot chrR}{\beta^2 \cdot chrP + chrR} \tag{12}$$

Where chrP (Character Precision) is the percentage of character n-grams in the system output that are also present in the reference, chrR (Character Recall) is the percentage of character n-grams in the reference that are present in the system output and β controls the weight given to recall versus precision; typically $\beta = 2$ is used (called chrF++) to emphasize recall.

3.9 Evaluation Process

Our evaluation process involves both component-level benchmarking in which we benchmarked segmentation, OCR and translation separately on distinct datasets tailored to each sub-task and end-to-end pipeline assessment using OpenMantra dataset.

3.9.1 OpenMantra

For the end-to-end pipeline evaluation, we utilised the OpenMantra dataset. Ideally suited for testing due to its specific scope, the dataset consists of 212 images across 5 manga volumes, with ground truth data generated by professional translators.

A key difference between OpenMantra and Manga109s is the annotation . OpenMantra provides annotations of the text inside a bubble. Each entry contains a bounding box for a single line of text, the corresponding Japanese characters, and the English translation. These entries are also pre-sorted by professional translator according to the correct reading order.

This structure necessitates the **Many-to-One** mapping strategy used in our evaluation. Since our segmentation model detects the entire speech bubble, a single prediction often encompasses multiple ground truth text boxes (lines). Therefore, evaluating the pipeline requires grouping these individual text lines into the predicted bubble region to reconstruct the full dialogue.

3.9.2 Component-Level Evaluation

Bubble & Panel Segmentation: is evaluated on the **Manga109s** test split (22 volumes). We report standard COCO metrics (Precision, Recall, mAP) to measure the geometric accuracy of bounding boxes and segmentation masks.

OCR: is evaluated on both **Manga109s** (cropped bubbles). We measure Character Error Rate (CER) and Word Error Rate (WER).

Sentence-Level Translation:
The baseline NMT models (ElanMT variants) and the LoRA [49] fine-tuned model were benchmarked on standard machine translation test sets: **Flores+** [50], **NTREX** [51], and **Opus100** [52]. These datasets provide diverse linguistic coverage but lack manga-specific context. Metrics include **BLEU, chrF++, BERTScore, and COMET**.

3.9.3 End-to-End Pipeline Evaluation

To assess the Context-Aware LLM and the integrated system (Segmentation → Ordering → Translation), we utilize the **OpenMantra** dataset. This is the only dataset providing coherent full-page contexts necessary for evaluating discourse-level phenomena.

Pipeline Matching Strategy:

To evaluate sequential outputs, predicted bubbles are matched to Ground Truth (GT) regions using a containment threshold of

$$\frac{\text{Area(Intersection)}}{\text{Area(GT)}} \geq 0.8$$

We employ this **containment-based** approach rather than standard Intersection-over-Union (IoU) to address granularity differences between detection and annotation. While our segmentation model predicts the entire speech balloon boundary, ground truth annotations often segment individual lines or text blocks separately.

This strategy enables a **Many-to-One** mapping capability. If a single predicted bubble contains multiple ground truth text regions (e.g., separate lines of text), all satisfying the containment threshold are assigned to that bubble. During evaluation, these matched text segments are sorted by their original index (preserving the reading order) and concatenated to form the complete reference text for that bubble.

- **Matched Pairs:** Predicted bubbles containing valid GT texts are evaluated for OCR and Translation accuracy.
- **Missed Detections:** GT text regions that are not contained by any predicted bubble are treated as omitted. As described in the metrics section, these are assigned empty strings for downstream evaluation.
- **False Positives:** Predicted bubbles that contain no ground truth text are flagged as empty predictions.

Ordering Metrics:
The layout analysis module is evaluated by comparing the sequence of matched GT indices against the true reading order using **Kendall's Tau (τ)** and **Exact Match Rate**.

Kendall's Rank Correlation Coefficient (τ_b)
To assess the correctness of the predicted reading order, we employ Kendall's Tau-b (τ_b), a robust correlation statistic that accounts for potential ties in the ranking sequence. Unlike simple accuracy, it penalizes the severity of ordering errors based on relative positions.

$$\tau_b = \frac{C - D}{\sqrt{(C + D + T_x)(C + D + T_y)}}$$

Where C is the number of concordant pairs (correctly ordered pair), D is the number of discordant pairs (incorrectly ordered pair), T_x is the number of ties in the predicted ranking and T_y is the number of ties in the ground truth ranking.
The coefficient ranges from -1 (perfect reversal) to $+1$ (perfect match). A high τ_b signifies that the narrative flow is preserved not just globally but also between local bubble pairs.

Exact Match Rate (EMR)
EMR measures the strict correctness of the layout analysis pipeline. A page is considered an exact match if and only if the entire sequence of predicted bubbles corresponds perfectly to the ground truth reading order ($\tau = 1$).

$$\text{EMR} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\{\tau_i = 1\}$$

Where N is the total number of pages evaluated and $\mathbb{1}$ is the indicator function.

Context-Aware Translation Metrics:

Unlike sentence-level tests, this evaluation measures translation quality **in situ**, taking into account text from previous and next bubbles. We report BLEU, BERTScore, and COMET scores computed on the matched speech bubbles.

Handling Detection Failures in Downstream Evaluation:
A critical consideration for end-to-end pipeline evaluation is the treatment of undetected speech bubbles. When the segmentation model fails to detect a ground-truth bubble, we assign an empty string ("") as the predicted OCR and translation output for that instance. This ensures that downstream metrics (CER, WER, BLEU, chrF++) naturally penalize detection failures through increased edit distances and reduced n-gram overlaps, rather than artificially inflating scores by evaluating only successfully detected bubbles.

4 Results and Evaluation

4.1 Bubble Segmentation

Table 6 and Table 7 illustrate bounding box and instance mask segmentation of four YOLO models: YOLOv8s, YOLOv8n, YOLOv11s, YOLOv11n.

Model	Precision	Recall	mAP50	mAP50-95
YOLOv8n	0.973	0.978	0.992	0.97
YOLOv8s	0.974	**0.982**	**0.993**	0.974
YOLOv11n	0.974	0.977	0.992	**0.979**
YOLOv11s	**0.977**	0.978	**0.993**	0.975
Liu et al. [10]	0.901	0.869	-	-
Rigaud et al. [7]	0.7221	0.8331	-	-

Table 6: Bubble bounding box detection performance of YOLO models

Method	Precision	Recall	mAP50	mAP50-95
YOLOv8n	0.974	0.979	0.993	0.916
YOLOv8s	0.975	**0.983**	0.993	**0.922**
YOLOv11n	0.975	0.979	0.993	0.92
YOLOv11s	**0.978**	0.979	0.993	0.921
Ngo et al. (2012) [16]	0.9467	0.7553	-	-
Rigaud et al. (2017) [7]	0.6227	0.6292	-	-
Dubray et al. (2019) [8]	0.9558	0.9404	-	-

Table 7: Bubble mask segmentation performance comparison

The results indicate that all four models achieve comparable performance, but overall, YOLOv8s demonstrates the most balanced and consistent performance across both bounding box detection and instance mask segmentation. Although our modern segmentation models show stronger performance than the methods reported in the comparison table, this should be interpreted with caution, as the evaluated methods are trained and tested on different datasets and target different segmentation objectives.

4.2 Panel Segmentation

Model	Precision	Recall	mAP50	mAP50-95
YOLOv8n	0.8666	0.8859	0.9286	0.6559
Xie et al. [53]	-	-	0.954	-

Table 8: Panel Segmentation performance of YOLOv8n

Table 8 illustrates the performance of YOLOv8n in panel segmentation. Comparing to the method that Xie et al. proposed, the gap is small but there is insufficient evidence to compare comprehensively, since both methods are trained and tested on different datasets.

4.3 OCR

To evaluate the OCR module, we tested the model on a subset of 640 speech bubbles using Character Error Rate (CER) and Word Error Rate (WER). As shown in Table 9, the raw model output yielded a CER of 0.1628. However, applying our deterministic post-processing pipeline significantly improved performance, ultimately reducing the CER to 0.1226 and WER to 0.4609. The step-wise breakdown reveals that **Unicode Normalization** provided the largest single gain (reducing CER by $\approx 2.2\%$) by correcting the model's tendency to generate full-width alphanumeric characters. Subsequently, **Punctuation Standardization** further reduced error rates by unifying inconsistent ellipsis styles, confirming that a significant portion of raw errors were formatting artifacts rather than recognition failures.

Table 9: Impact of post-processing operations on OCR performance

Step	**Operation**	**CER**	**WER**
0	Raw Output	0.1628	0.5984
1	Unicode Normalize	0.1408	0.4984
2	Whitespace Removal	0.1408	0.4984
3	Ellipsis Fixing	0.1231	0.4625
4	**Final Pipeline**	**0.1226**	**0.4609**

4.4 Translation

Table 10: Performance Comparison of GoogleTranslator, Elan, Qwen2.5-0.5B, and Qwen2.5-1.5B Models

Model	BLEU	SacreBLEU	chrf	chrF++	BERTScore F1	COMET
GoogleTranslator-ja-en	0.0948	**0.1598**	**0.3534**	**0.3262**	0.5831	0.7378
Elan Base	0.0261	0.0719	0.2123	0.1848	0.5586	0.591
Elan BT	0.047	0.0978	0.2649	0.2441	0.5783	0.6475
Elan Tiny (default)	0.0386	0.0895	0.2551	0.2344	0.5805	0.6374
Qwen2.5-0.5B-Instruct (finetuned, context_window = 5)	0.0497	0.0839	0.2747	0.2581	0.5813	0.6413
Qwen2.5-0.5B-Instruct (finetuned, context_window = 4)	0.0556	0.0910	0.2836	0.2663	0.5820	0.6445
Qwen2.5-0.5B-Instruct (finetuned, context_window = 3)	0.0482	0.0836	0.2836	0.2662	0.5822	0.6354
Qwen2.5-0.5B-Instruct (finetuned, context_window = 2)	0.0497	0.0843	0.2789	0.2618	0.5802	0.6396
Qwen2.5-0.5B-Instruct (finetuned, context_window = 1)	0.0498	0.0797	0.2788	0.2608	0.5782	0.6319
Qwen2.5-1.5B-Instruct (finetuned, context_window = 5)	0.0618	0.098	0.2905	0.2721	0.6042	0.6813
Qwen2.5-1.5B-Instruct (finetuned, context_window = 4)	0.05282	0.0986	0.2868	0.2685	0.6027	0.683
Qwen2.5-1.5B-Instruct (finetuned, context_window = 3)	0.0621	0.1014	0.2939	0.2749	0.603	0.6899
Qwen2.5-1.5B-Instruct (finetuned, context_window = 2)	0.0623	0.0995	0.2944	0.2756	0.6044	0.6905
Qwen2.5-1.5B-Instruct (finetuned, context_window = 1)	0.0598	0.1021	0.2903	0.2719	**0.6049**	0.6914
2+2 [5]	0.2114	-	-	-	-	0.779
Scene-NMT [5]	0.2102	-	-	-	-	0.782
Scene-aware NMT [26]	**0.2146**	-	-	-	-	**0.79**

Obtained results from Towards Fully Automated Manga Translation [5] were re-implemented by Kaino et al. [26] in their publication "Utilizing Longer Context than Speech Bubbles in Automated Manga Translation". We observe a performance gap compared to previously reported results. We attribute this mainly to differences in data distribution and the absence of domain-specific tuning, which we leave for future work.

5 Desktop Application

5.1 Architecture Overview

The desktop application `Lucile` is designed as a modular graphical interface that acts as an abstract layer for a multi-stage machine learning pipeline. The application is built using Python and PySide6 framework, the official Python binding for the Qt project.
The system uses the data by sharing a single object during runtime. The `AppData` component stores `image_path`, `Image` itself, and the results from each phase across the application's lifecycle.

5.2 Use Case Specification

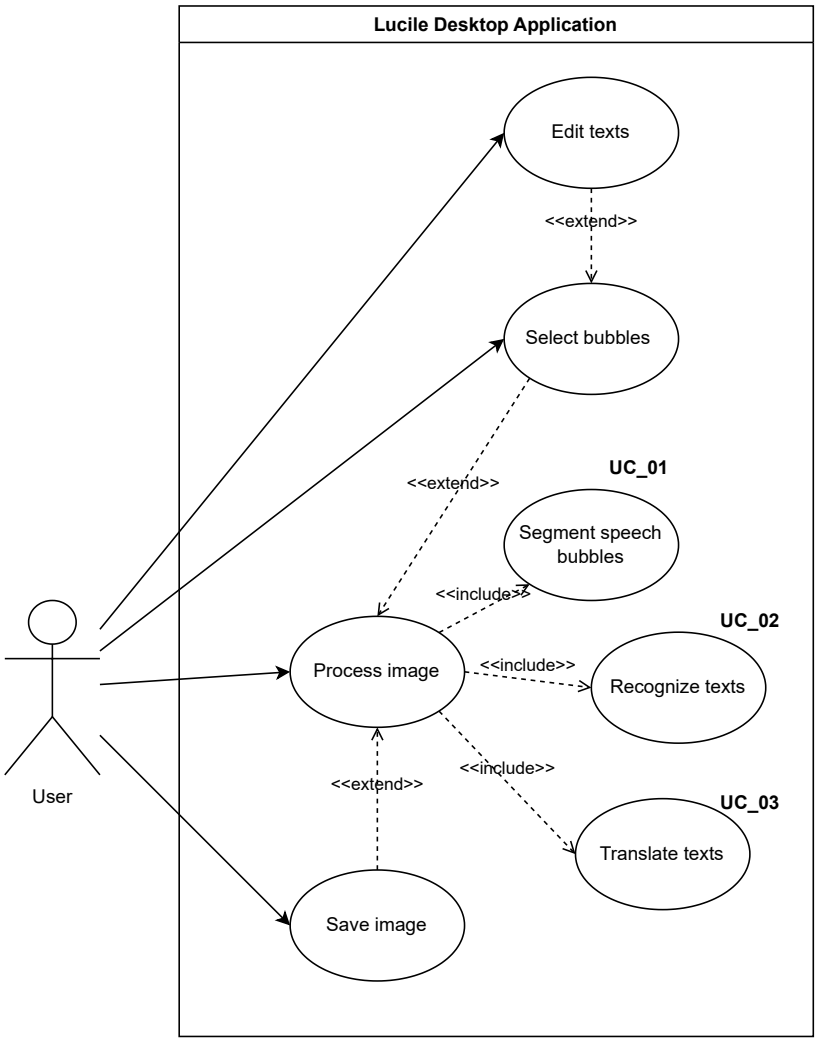


Figure 11: Use Case Diagram of the Manga Translation System

Figure 11 illustrates the use case diagram of our Manga Translation System, showing the main actor and the core functionalities provided by the system. Based on this diagram, each identified use case is analyzed in detail in the following sections.

5.2.1 UC_01: Segment speech bubbles

Segment speech bubbles	
Actors	System
Description	System automatically detects, segments and splits speech bubbles from the input image.
Data	An input image
Stimuli	User drags the image into queue.
Response	The system processes the input image and displays the image with segmented speech bubbles.
Pre-conditions	An image is dragged into the queue. Segmentation model is loaded.
Post-conditions	The image is displayed with bounded speech bubbles. System performs text recognition.

5.2.2 UC_02: Recognize texts

Recognize texts	
Actors	System
Description	System detects Japanese texts utilizing OCR model. The recognized texts are stored with no visual output.
Data	An input image
Stimuli	User drags the image into queue.
Response	Recognized Japanese texts are stored internally and can be viewed by user selecting segmented bubbles.
Pre-conditions	An image is dragged into the queue. OCR model is loaded.
Post-conditions	System proceeds to translation stage. Recognized Japanese texts are available for user editing.

5.2.3 UC_03: Translate texts

Translate texts	
Actors	System
Description	System translates recognized Japanese texts into English using translation model.
Data	An input image
Stimuli	User drags the image into queue.
Response	The system translates the recognized Japanese texts into English and displays the final image with translated English texts rendered into speech bubbles.
Pre-conditions	An image is dragged into the queue.
Post-conditions	The image is displayed with bounded speech bubbles and the corresponding translated text shown above each region.

5.3 Core Data Structures

The AppData class is an object that holds the global state, including `pil_image`, `image_path`, and a list of `Bubble` objects. This allows each component to access the image the moment it is loaded in the Segmentation module.

The `Bubble` class is an object that holds specific detection instances. It holds the geometry of a bubble including the segmentation mask in `QPolygonF` and the bounding box `bbox`. The text recognized by the OCR tab and the corresponding English translation are also stored here. In the UI, these data objects are linked to visual elements using Qt's UserRole data storage, allowing direct object retrieval.

5.4 Functional Components

The application workflow is divided into four sequential stages, each managed by a dedicated controller class inheriting from QWidget.

5.4.1 Segmentation Module

`SegmentBubbleTab` manages the initial image acquisition and object detection. In this tab, the images are loaded via `PIL` for backend processing and converted to `QPixmap` for frontend rendering in the `QGraphicsScene`. The module wraps the Ultralytics YOLO model. The users can load custom weights from a `.pt` file and adjust the confidence threshold slider to tune sensitivity dynamically. The resulting detection masks are converted into QGraphicsPolygonItem overlays. The system binds the visual polygon to the logical `Bubble` object by using `setData(key, value)`. This linkage allows the system to identify exactly which data object to remove from `AppData` when a user selects a polygon and triggers the delete command.

5.4.2 OCR Module

The `OCRTab` handles text extraction using the `MangaOCR` model from `manga-ocr`. The system iterates through the valid `Bubble` objects, using their stored bounding boxes to crop specific regions from the source image for inference. The results are displayed in a `QListWidget`. To prevent UI clutter, the preview text is truncated to 20 characters with ellipses. A `QTextEdit` widget allows the user to correct the OCR output. The application managed the event loop by blocking signals during updates to ensure that the edits in the textbox immediately update both the `Bubble` data object and the visual text overlay on the canvas without causing recursive event loops.

5.4.3 Machine Translation Module

The `TranslateTab` handles translation of text using the `ElanMT` transformer model. The model is lazily loaded when requested. Three models are available: **tiny**, **base**, **bt**. The interface shows a split view, a list of bubbles, and the read-only Japanese source text with their editable field for translated text.

5.4.4 Typesetting Module

The `ReplaceTab` handles typesetting. It replaces the source text in bubbles with translated text using the `MangaTypesetter` backend. Because the backend typesetter requires binary masks rather than vector polygons, the system implements a helper function, `_polygon_to_mask`, which rasterizes the `QPolygon` objects into Numpy arrays using `cv2.fillPoly`. The engine paints a white background and renders the translated text over the original image using the generated masks. The final result is converted from a raw byte array back into a QImage for display and export to disk.

5.5 Exception Handling

All machine learning inference calls and file input/output (I/O) operations are encapsulated in try-except blocks. A custom dialog class, `MessageDialog` was implemented to handle runtime errors gracefully. Instead of crashing the application to the console, errors such as missing model files or inference failures are displayed in a pop up window.

6 Conclusion and Future Work

In this work, we presented an end-to-end pipeline for automatic Japanese manga translation, integrating speech bubble segmentation, optical character recognition, machine translation, and

typesetting into a unified Qt-based desktop application. Our system leverages YOLOv8 for bubble and panel segmentation, MangaOCR for Japanese text recognition with post-processing, and Qwen2.5-1.5B-Instruct fine-tuned with LoRA for context-aware translation. Rather than replacing human translators, the application aims to reduce repetitive manual effort while providing editing functionality at each stage of the workflow.

Future work includes improving OCR accuracy through expanded training data and enhanced post-processing techniques, developing more sophisticated context-aware translation that better captures conversation history and speaker identity across dialogue sequences, and refining the application interface to provide a more intuitive experience with additional features tailored to professional translator workflows.

Acknowledgements

We would like to express our sincere gratitude to our supervisor, Assoc. Prof. Tran Giang Son, for his invaluable guidance, support, and constructive feedback throughout the duration of this project.

We also gratefully acknowledge ICTLab for providing access to the server resources used in our experiments. In addition, we would like to thank the Aizawa–Yamasaki–Matsui Laboratory for granting us permission to use their dataset, which was essential to this research.

References

[1] M. T. Team, "Manga translator," https://www.mangatranslate.com/#faq.

[2] C. T. Team, "Comic translate," https://github.com/ogkalu2/comic-translate, 2024.

[3] M. E. Team, "Mantra - manga translator," Mobile application, Google Play, accessed: Dec. 2025.

[4] S. Saikumar and R. Raghavendra, "Automatic translation of mangas into other languages," *International Journal of Creative Research Thoughts (IJCRT)*, vol. 10, no. 4, pp. c511–c517, 2022. [Online]. Available: https://ijcrt.org/papers/IJCRT2204293.pdf

[5] R. Hinami, S. Ishiwatari, K. Yasuda, and Y. Matsui, "Towards fully automated manga translation," in *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*. Association for the Advancement of Artificial Intelligence, 2021.

[6] C. Rigaud, J.-C. Burie, J.-M. Ogier, D. Karatzas, and J. van de Weijer, "An active contour model for speech balloon detection in comics," in *Proceedings of the 2013 12th International Conference on Document Analysis and Recognition (ICDAR)*, 2013, pp. 1240–1244. [Online]. Available: https://hal.science/hal-00879816/file/2013_Rigaud_An_active_contour_model_for_speech_balloon_detection_in_comics.pdf

[7] C. Rigaud, J.-C. Burie, and J.-M. Ogier, "Text-independent speech balloon segmentation for comics and manga," in *International Workshop on Graphics Recognition*. Springer, Lecture Notes in Computer Science, 2017.

[8] D. Dubray and J. Laubrock, "Deep cnn-based speech balloon detection and segmentation for comic books," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 2019, pp. 1237–1243.

[9] T. Melistas, G. Siglidis, F. Kalogiannis, and I. Manouach, "A deep learning pipeline for the synthesis of graphic novels," in *Proceedings of the 12th International Conference on Computational Creativity (ICCC)*. México City, Mexico: Association for Computational Creativity, 2021, pp. 256–265.

[10] X. Liu, Y. Wang, and Z. Tang, "A clump splitting based method to localize speech balloons in comics," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, 2015, pp. 901–905.

[11] S. Ranjini and M. Sundaresan, "Text extraction from digital english comic image using two blobs extraction method," in *2012 International Conference on Computing, Communication and Applications (ICCCA)*, 2012, pp. 1–5.

[12] Y. Aramaki, Y. Matsui, T. Yamasaki, and K. Aizawa, "Text detection in manga by combining connected-component-based and region-based classifications," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 2901–2905.

[13] N. S. T. Hirata, I. S. Montagner, and R. J. Hirata, "Comics image processing: Learning to segment text," in *Proceedings of the ACM International Conference on Multimedia*. Association for Computing Machinery, 2016, pp. 1–6.

[14] W.-T. Chu and C.-C. Yu, "Text detection in manga by deep region proposal, classification, and regression," in *2018 IEEE Visual Communications and Image Processing (VCIP)*. Taichung, Taiwan: IEEE, 2018, pp. 1–4.

[15] G. Soykan, D. Yuret, and T. M. Sezgin, "A Comprehensive Gold Standard and Benchmark for Comics Text Detection and Recognition," in *Document Analysis and Recognition – ICDAR 2024 Workshops*, ser. Lecture Notes in Computer Science. Springer Nature Switzerland, 2024, vol. 14935, pp. 168–197. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-70645-5_12

[16] A. K. Ngo, J.-C. Burie, and J.-M. Ogier, "Panel and speech balloon extraction from comic books," in *2012 10th International Conference on Document Analysis and Recognition*. IEEE, 2012, pp. 1163–1167.

[17] 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*, ser. MM '14. New York, NY, USA: Association for Computing Machinery, 2014, pp. 1125–1128.

[18] N. V. Nguyen, C. Rigaud, and J.-C. Burie, "What do we expect from comic panel extraction?" in *2019 2nd International Conference on Document Analysis and Recognition Workshops (ICDARW)*, 2019, pp. 44–49.

[19] Y. Omori, K. Nagamizo, and D. Ikeda, "Algorithms for estimation of comic speakers considering reading order of frames and texts," in *2022 12th International Congress on Advanced Applied Informatics (IIAI-AAI)*. IEEE, 2022, pp. 367–372, iEEE Xplore.

[20] M. Nagao, J.-i. Tsujii, and J. Nakamura, "Machine translation from japanese into english," *Proceedings of the IEEE*, vol. 74, no. 7, pp. 993–1012, 1986.

[21] S. Zhou, T. Zhou, B. Wei, Y. Luo, Y. Mu, Z. Zhou, C. Wang, X. Zhou, C. Lv, Y. Jing, L. Wang, J. Zhang, C. Huang, Z. Yan, C. Hu, B. Li, T. Xiao, and J. Zhu, "The niutrans machine translation systems for wmt21," in *Proceedings of the Sixth Conference on Machine Translation (WMT)*. Association for Computational Linguistics, 2021, pp. 1007–1015.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017, pp. 5998–6008. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[23] Q. Wang, B. Li, T. Xiao, J. Zhu, C. Li, D. F. Wong, and L. S. Chao, "Learning deep transformer models for machine translation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, pp. 1810–1822. [Online]. Available: https://aclanthology.org/P19-1176/

[24] M. X. Chen, O. Firat, A. Bapna, M. Johnson, W. Macherey, G. Foster, L. Jones, M. Schuster, N. Shazeer, P. Niki, A. Vaswani, J. Uszkoreit, L. Kaiser, Z. Chen, Y. Wu, and M. Hughes, "The best of both worlds: Combining recent advances in neural machine translation," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, 2018, pp. 76–86. [Online]. Available: https://aclanthology.org/P18-1008/

[25] K. Kinugawa, H. Mino, I. Goto, and N. Shirai, "Findings of the wmt 2024 shared task on non-repetitive translation," in *Proceedings of the Ninth Conference on Machine Translation (WMT 2024)*. Miami, Florida, USA: Association for Computational Linguistics, 2024, pp. 715–727. [Online]. Available: https://www2.statmt.org/wmt24/pdf/2024.wmt-1.60.pdf

[26] H. Kaino, S. Sugihara, T. Kajiwara, T. Ninomiya, J. B. Tanner, and S. Ishiwatari, "Utilizing longer context than speech bubbles in automated manga translation," in *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*. Torino, Italy: ELRA and ICCL, 2024, pp. 17 337–17 342. [Online]. Available: https://aclanthology.org/2024.lrec-main.1505

[27] U.-R. Ko and H.-G. Cho, "Sickzil-machine: A deep learning based script text isolation system for comics translation," in *Document Analysis Systems – DAS 2020*, ser. Lecture Notes in Computer Science. Springer, 2020, vol. 12116, pp. 413–425. [Online]. Available: https://gwern.net/doc/ai/anime/danbooru/2020-ko.pdf

[28] K. Aizawa, A. Fujimoto, A. Otsubo, T. Ogawa, Y. Matsui, K. Tsubota, and H. Ikuta, "Building a manga dataset "manga109" with annotations for multimedia applications," *IEEE MultiMedia*, vol. 27, no. 2, pp. 8–18, 2020.

[29] Y. Matsui, K. Ito, Y. Aramaki, A. Fujimoto, T. Ogawa, T. Yamasaki, and K. Aizawa, "Sketch-based manga retrieval using manga109 dataset," *Multimedia Tools and Applications*, vol. 76, no. 20, pp. 21 811–21 838, 2017.

[30] R. Varghese and S. M., "Yolov8: A novel object detection algorithm with enhanced performance and robustness," in *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, 2024, pp. 1–6.

[31] G. Jocher and J. Qiu, "Ultralytics yolo11," https://github.com/ultralytics/ultralytics, 2024.

[32] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 1571–1580.

[33] C.-Y. Wang, H.-Y. M. Liao, and I.-H. Yeh, "Designing network design strategies through gradient path analysis," 2022. [Online]. Available: https://arxiv.org/abs/2211.04800

[34] P. Hidayatullah, N. Syakrani, M. R. Sholahuddin, T. Gelar, and R. Tubagus, "Yolov8 to yolo11: A comprehensive architecture in-depth comparative review," *arXiv preprint*, vol. abs/2501.13400, 2025, preprint. [Online]. Available: https://arxiv.org/abs/2501.13400

[35] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759–8768.

[36] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 936–944.

[37] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," 2021. [Online]. Available: https://arxiv.org/abs/2107.08430

[38] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: Real-time instance segmentation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9156–9165.

[39] R. Smith, "An overview of the tesseract ocr engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2. Curitiba, Brazil: IEEE, 2007, pp. 629–633. [Online]. Available: https://ieeexplore.ieee.org/document/4376991

[40] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 347–10 357.

[41] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[42] M. Junczys-Dowmunt, R. Grundkiewicz, T. Dwojak, H. Hoang, K. Heafield, T. Neckermann, F. Seide, U. Germann, A. F. Aji, N. Bogoychev, A. F. T. Martins, and A. Birch, "Marian: Fast neural machine translation in c++," 2018. [Online]. Available: https://arxiv.org/abs/1804.00344

[43] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, E. Blanco and W. Lu, Eds. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 66–71. [Online]. Available: https://aclanthology.org/D18-2012/

[44] M. Rikters, R. Ri, T. Li, and T. Nakazawa, "Designing the business conversation corpus," in *Proceedings of the 6th Workshop on Asian Translation*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 54–61. [Online]. Available: https://www.aclweb.org/anthology/D19-5204

[45] R. Rei, J. G. C. de Souza, D. M. Alves, C. Zerva, A. C. Farinha, T. Glushkova, A. Lavie, L. Coheur, and A. F. T. Martins, "Comet-22: Unbabel-ist 2022 submission for the metrics shared task," in *Proceedings of the Seventh Conference on Machine Translation (WMT)*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, 2022, pp. 578–585. [Online]. Available: https://aclanthology.org/2022.wmt-1.52/

[46] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," in *International Conference on Learning Representations (ICLR)*, 2020. [Online]. Available: https://openreview.net/forum?id=SkeHuCVFDr

[47] M. Popović, "chrf: character n-gram f-score for automatic mt evaluation," in *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Lisbon, Portugal: Association for Computational Linguistics, 2015, pp. 392–395. [Online]. Available: https://aclanthology.org/W15-3049/

[48] ——, "chrF++: words helping character n-grams," in *Proceedings of the Second Conference on Machine Translation*, O. Bojar, C. Buck, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, and J. Kreutzer, Eds. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 612–618. [Online]. Available: https://aclanthology.org/W17-4770/

[49] E. J. Hu, S. Yelong, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=nZeVKeeFYf9

[50] NLLB Team, M. R. Costa-jussà, J. Cross, O. Çelebi, M. Elbayad, K. Heafield, K. Heffernan, E. Kalbassi, J. Lam, D. Licht, J. Maillard, A. Sun, S. Wang, G. Wenzek, A. Youngblood, B. Akula, L. Barrault, G. M. Gonzalez, P. Hansanti, J. Hoffman, S. Jarrett, K. R. Sadagopan, D. Rowe, S. Spruit, C. Tran, P. Andrews, N. F. Ayan, S. Bhosale, S. Edunov, A. Fan, C. Gao, V. Goswami, F. Guzmán, P. Koehn, A. Mourachko, C. Ropers, S. Saleem, H. Schwenk, and J. Wang, "Scaling neural machine translation to 200 languages," *Nature*, vol. 630, no. 8018, pp. 841–846, 2024. [Online]. Available: https://doi.org/10.1038/s41586-024-07335-x

[51] C. Federmann, T. Kocmi, and Y. Xin, "NTREX-128 – news test references for MT evaluation of 128 languages," in *Proceedings of the First Workshop on Scaling Up Multilingual Evaluation*. Online: Association for Computational Linguistics, nov 2022, pp. 21–24. [Online]. Available: https://aclanthology.org/2022.sumeval-1.4

[52] B. Zhang, P. Williams, I. Titov, and R. Sennrich, "Improving massively multilingual neural machine translation and zero-shot translation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds. Online: Association for Computational Linguistics, Jul. 2020, pp. 1628–1639. [Online]. Available: https://aclanthology.org/2020.acl-main.148

[53] M. Xie, J. Lin, H. Liu, C. Li, and T.-T. Wong, "Advancing manga analysis: Comprehensive segmentation annotations for the manga109 dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025, pp. 8869–8878.

7 Appendix


```
{
  "model": "unsloth/Qwen2.5-1.5B-Instruct",
  "training_file": [
    "/home/zendragonxxx/grprj/group-project-b3/data/translation/pretrain/data/pretrain-data.jsonl",
    "/home/zendragonxxx/grprj/group-project-b3/data/translation/bsd_ja_en/training/train_full_context.jsonl"
  ],
  "finetuned_model_id": "TheBlindMaster/Qwen2.5-1.5B-Instruct-emergent-finetune-train_full_context-all-full-r64-e1",
  "max_seq_length": 861,
  "load_in_4bit": false,
  "is_peft": true,
  "loss": "sft",
  "target_modules": [
    "q_proj",
    "k_proj",
    "v_proj",
    "o_proj",
    "gate_proj",
    "up_proj",
    "down_proj"
  ],
  "layers_to_transform": null,
  "r": 64,
  "lora_alpha": 128,
  "lora_dropout": 0.0,
  "use_rslora": true,
  "lr_scheduler_type": "cosine",
  "learning_rate": 0.0001,
  "per_device_train_batch_size": 8,
  "gradient_accumulation_steps": 16,
  "warmup_steps": 100,
  "optim": "adamw_8bit",
  "weight_decay": 0.01,
  "epochs": 1,
  "push_to_private": true,
  "merge_before_push": true,
  "train_on_responses_only": true,
  "evaluation_steps": 2000,
  "save_steps": 4000,
  "early_stopping_patience": 5
}
```


Figure 12: Training configuration for **Qwen2.5-1.5B-Instruct** using LoRA