Automatic Question Generation from Handwritten Lecture Notes Using TrOCR Text Recognition and T5 Language Processing

by

Rommel John H. Ronduen
Jan Adrian C. Manzanero

A Thesis Report Submitted to the School of Electrical, Electronics, and Computer Engineering in Partial Fulfilment of the Requirements for the Degree

Bachelor of Science in Computer Engineering

Mapúa University
July 2025

APPROVAL SHEET

This is to certify that I have supervised the preparation of and read the thesis paper prepared by **Rommel John H. Ronduen** and **Jan Adrian C. Manzanero** entitled **Automatic Question Generation from Handwritten Lecture Notes Using TrOCR Text Recognition and T5 Language Processing** and that the said paper has been submitted for final examination on May 15, 2025, by the Oral Examination Committee.

Analyn N. Yumang
Thesis Adviser

As members of the Oral Examination Committee, we certify that we have examined the paper and hereby recommend that it be accepted as fulfillment of the thesis requirement for the Degree Bachelor of Science in Computer Engineering.

Jocelyn F. Villaverde
Panel Member

Meo Vincent C. Caya
Panel Member

Noel B. Linsangan
Committee Chairman

This thesis paper is hereby approved and accepted by the School of EECE as fulfillment of the thesis requirement for the Degree **Bachelor of Science in Computer Engineering**.

Arnold C. Paglinawan
Dean, School of EECE

ACKNOWLEDGEMENT

The researchers would like to express their gratitude to those who provided support and assistance throughout this study. First and foremost, they sincerely thank their parents for their unwavering support and sacrifices, which made this research possible.

Additionally, they extend their most profound appreciation to their advisor, Engr. Analyn N. Yumang, for her guidance and encouragement throughout the research. Her mentorship played a vital role in shaping this paper, and they are thankful for the opportunity to have worked under her supervision.

They also acknowledge their panelists, Engr. Noel B. Linsangan, Dr. Jocelyn F. Villaverde, and Engr. Meo Vincent C. Caya, for their valuable feedback and suggestions, which significantly contributed to enhancing this study.

TABLE OF CONTENTS

LIST OF TABLES

Main Paper

Article 1

Article 2

LIST OF FIGURES

Main Paper

Article 1

Article 2

ABSTRACT

This research involves the creation and evaluation of a system that allows for text extraction and automatic question generation (AQG) using a Text-to-Text transformer (T5) and Transformer-based Optical Character Recognition (TrOCR) pipeline. It addresses the lack of OCR implementation with AQG alongside handwritten notes as the basis for questions generation. With the use of a Raspberry Pi 5, web camera, and a touchscreen display, factoid-type questions are created from image captures of single-column handwritten notes that only contain textual information. The T5 large language model (LLM) used was fine-tuned using the Stanford Question Answering Dataset (SQuAD) for facilitating question generation. Evaluation was done using the word error rate (WER) for OCR, while the Recall Oriented Understudy for Gisting Evaluation (ROUGE) and Bilingual Language Evaluation Understudy (BLEU) metrics were used for AQG. The system had a WER of 0.40, a ROUGE- 1 score of 0.358, and a question validity rate of 68% that led to questions with decent context similarity versus student-made questions. It is recommended that advanced hardware is to be used, consequently allowing for larger versions of T5 and TrOCR. Moreover, much emphasis should be put on the OCR preprocessing.

Chapter 1

INTRODUCTION

Handwritten lecture notes are considered the standard method for capturing and facilitating learning among students [1]. A relevant percentage of students still prefer handwriting lecture notes to internalize information received from lessons [2]. These learning artifacts contain valuable information for learners and may serve as the basis for other forms of learning materials. One such concept is the use of review questions in the form of quizzes, test banks, and flashcards, which can be tedious when compared to handwritten sources. With the advent of artificial intelligence (AI), education has become one of the primary beneficiaries, thanks to its aid in creating learning materials. The creation of review questions is involved in this generative AI through a process defined in the literature as automatic question generation (AQG). This research aims to facilitate AQG by utilizing handwritten lecture notes as a source for generating questions. This study uses a fine-tuned T5 language model and the base TrOCR handwritten model on the Raspberry Pi 5. Context indexing is achieved using the spaCy software library, along with the Rapid Automatic Keyword Extraction (RAKE) algorithm. With the aid of the Gemini model for text enhancement, AQG from handwritten lecture notes has become a possibility using web camera captures of the given notes.

AQG systems have been implemented using various digital media types. [3] utilized video lectures for AQG wherein an automatic speech recognition algorithm by Google extracted text from the audio feed and generated question-answer pairs using the Bidirectional and Auto-Regressive Transformer (BART) LLM. [4] utilized text files as a source for creating

Python programming questions using the BERT ERT model for extracting keywords and phrases, and T5 for the generation of questions. Several more variants of the T5 model have also been discussed in [13], noting that it is the most predominant LLM in use for question generation systems. Despite the prevalence of AQG systems, considerable work remains to be done in extracting data from handwritten information [5].

Typical AQG systems expect data inputs to come from digital sources, not physical sources. No implementation processes handwritten lecture notes directly for the AQG process, as advancements in the research context assumed that data inputs are in digital form, possibly through image files or manually encoded text in a spreadsheet format reflecting a physical source. Instead of addressing these issues, there is a need for a system that immediately captures the physically written text, where a text extraction feature gathers and processes the text and consequently utilizes AQG for the generation of questions and answers. Therefore, this research proposal aims to bridge these gaps by using the Transformer-based Optical Character Recognition (TrOCR) transformer and its interface with the T5 transformer, enabling AQG.

In general, the goal of this research is to perform AQG from physical handwritten notes using TrOCR text recognition and T5 language processing. Specifically, this research seeks to achieve these objectives: (1) to be able to extract text from handwritten lecture notes using TrOCR text recognition and refine said text using Gemini 1.5 Flash LLM; (2) to be able to utilize a fine-tuned base version of T5 on SQuAD for AQG on indexed keywords from spaCy and Rapid Automatic Keyword Extraction (RAKE) on the refined text; (3) to be able to utilize a Raspberry Pi 5 within a constructed enclosure with illumination for facilitating the system processes and experimental set-up; and (4) to be able to evaluate the system using the word

error rate (WER) for the OCR, and the ROUGE and BLEU metrics for the comparison of the AQG results to student-generated questions that the researchers will accumulate.

The results of this research may aid in further implementing AI abilities in real-world learning methods. More specifically, this study endeavors to expand the application of natural language processing (NLP) beyond its purely digital scope. Addressing the specific limitation of AQG in processing immediate digital information, students and educators no longer need to transform handwritten lecture notes into processable data forms, such as images or text data. The process of transformation involves manual input into word-processing software applications, which can be automated through the study's incorporation of OCR provided by a transformer network, as well as the AQG, utilizing the T5 language model.

The system developed in this research is limited to performing AQG on handwritten lecture notes written using the standard English alphabet on plain letter-size paper with dimensions of 8.5 inches by 10.5 inches. The lecture notes must be single-column and must not contain diagrams, equations, symbols, or other special characters. It should be noted that this system utilizes the base versions of the T5 and TrOCR models, where the former uses SQuAD for the fine-tuning and Gemini 1.5 Flash for the text correction and context completion of OCR-extracted text, as well as the spaCy-RAKE keyword extraction for AQG basis. Additionally, the system's operation is limited by the processing capability of the 8-gigabyte model of the Raspberry Pi 5. A different model for OCR or AQG may yield differing results, as well as for the text enhancement model. Lastly, the use of alternative hardware may produce different results. For the system evaluation, the evaluation dataset will consist of handwritten lecture notes and questions about alternative hardware that may be generated from an

undergraduate cybersecurity class. WER will be utilized for the evaluation of OCR, while ROUGE and BLEU will be used for the assessment of AQG.

Chapter 2

REVIEW OF RELATED LITERATURE

Automatic question generation

AQG is defined as the process in which questions are generated from an arbitrary source [5]. AQG may consist of keyphrase extraction, sentence parsing, and question evaluation. It is during the extraction and parsing process that AQG systems differ. It was realized that these existing AQG implementations did not incorporate OCR for extracting handwritten notes as a source for question generation. For instance, [3] developed a system that utilized video as a source for questions requiring audio-to-text processing, employing the Bidirectional and Auto-regressive Transformer (BART) model, which achieved 80% context similarity and proper accuracy and fluency for the questions. Another implementation involved manually inputting questions in English education. Here, question generation involved part-of-speech (POS) tagging through AllenNLP and rule-based templates for creating questions, which achieved a precision of approximately 60-70%. On a similar use case to AQG, [6] also utilized text input sources to generate summaries and answer questions using BART, which achieved a ROUGE score of 0.400. Another AQG use case involved the intake of programming source code in manual text format as a source for rule-based question generation through keyword extraction methods (Gaur). An interesting implementation utilized the T5 transformer to generate factoid-type questions through keyword extraction using the Bidirectional Encoder Representation from Transformers (BERT) model on text files [6]. The researchers also utilized SQuAD for training the T5 model. Similarly, [10] utilized T5 and SQuAD for generating factoid-type questions that expected general paragraphs from

manual text inputs, leading to 75% to 82% semantic similarity using BERT analysis. Generating factoid-type questions through keyword extraction using the Bidirectional Encoder Representation from Transformers (BERT) model from text files [6]. The researchers also utilized SQuAD for training the T5 model. Similarly, [10] utilized T5 and SQuAD for generating factoid-type questions that expected general paragraphs from manual text inputs, leading to 75% to 82% semantic similarity using BERT analysis.

Ultimately, the implementations of AQG utilized a paradigm involving a model for generating questions and an algorithm for scanning through relevant text from input sources, including video, text files, or manual input. No utility of OCR was presented, as well as the usage of handwritten lecture notes as the source for question generation. As such, this research aims to utilize an OCR framework to extract text from handwritten notes and use it for the generation of questions. The researchers of this study intend to utilize T5 due to its fine-tuning capability with SQuAD, which allows for exact text string inputs in the usage and fine-tuning of T5, as demonstrated in the studies of [6] and [10].

Review questions for AQG.

Review questions have been considered a standard learning material among students, as they facilitate learning, especially in this digital age [6] [7]. Frequent assessment using these questions has become a common tool for pedagogical standards, which drives the need for such questions. Questions can be primarily classified as subjective (based on opinions) or objective (based on facts). An article by Arbaaeen and Shah delves into the nuances of objective questions into six main categories: (1) factoid-type questions, which require a factual answer based on given information and always begin with the wh-phrases, (2) listed or enumerated questions, which require a list of facts as answers, (3) confirmation questions,

which require a “yes” or a “no” answer, (4) causal questions, which require a detailed answer describing a cause of an entity or an event, (5) hypothetical questions, which require information associated with a hypothetical event, and (6) complex questions, which require numerous references to gather tremendous data to compose an answer [5]. Examples of these categories are provided in Table 2.1. Much focus is to be given to factoid questions, which are generally initiated by the WH phrases — what, who, where, when, etc. Literature for AQG systems focuses on generating factoid questions, as they have fact-based answers, allowing systems to formulate these questions from at least one sentence [6]

Question Type		**Example**
Subjective Questions	Opinion-Based	Do you want to live on the moon?
Objective Questions	Factoid-Type	What color is the moon?
	List-Type	List all the moons of the solar system.
	Confirmation	Is the Sun a star?
	Causal	How do the planets orbit the Sun?
	Hypothetical	What if the Sun explodes?

Table 2.1: Question classification table (sourced from [5] **and** [6]**)**

The model architecture of the T5 Transformer

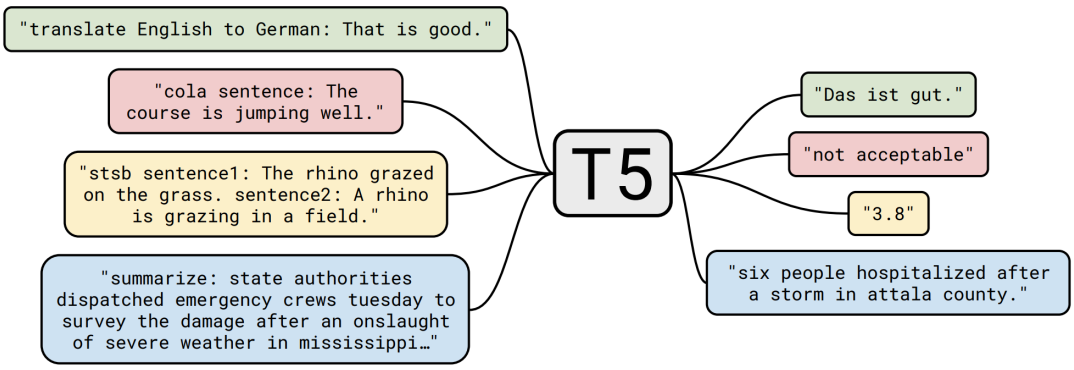


Figure 2.1: T5 model framework (sourced from [9]**)**

Relevant in the field of NLP is the introduction of transformer-based models, which can analyze and process entire sentences at once. Because of this, transformer-based models enable NLP applications — such as question generation — that are more powerful and efficient compared to the traditional word-by-word analysis of sequence-based models like RNN and LSTM [8]. The T5 LLM created by Google is one of these transformer-based LLMs, designed as a successor to BERT [9].

Malhar et al. developed an AQG system that takes input from a selected group of words using a T5 model that has undergone fine-tuning with a custom dataset. They noted that a properly fine-tuned T5 model allows for the inclusion of text elements into other text elements in the form of questions [10]. It is said that the manual extraction of phrases served as the basis of the AQG, not the data utilized for fine-tuning. Moreover, the T5 model can also be used in conjunction with an input pipeline. Chelliah et al. demonstrated the T5 model's capabilities by using it in a pipeline that extracted text data from digital documents, serving as input to a fine-tuned T5 model that generated questions from chosen contexts [8].

Ultimately, the consensus framework for utilizing T5 alongside an arbitrary input pipeline was realized, as depicted in the figure above. It was discovered that AQG is indeed possible using the T5 model and a specialized dataset. However, this model only works with the inclusion of user intervention, not only for the input pipeline, but also for the processing, which involves some manual extraction done on the given context to direct the model in creating the relevant questions. As such, the researchers plan to offer a successor to the current consensus architecture of T5 AQG by eliminating the need for manual context extraction and modification, allowing another system to perform, and ensuring a seamless user experience.

In the existing literature regarding AQG, it has been noted that the challenge revolves around the inclusion of information-gathering processes, where the need for an existing dataset is utilized as a heuristic for creating the desired question-answer pairs. Moreover, many of the systems developed were implemented to process immediate digital information, i.e., the inputs are assumed to be digital and in a specific format. Some of the systems considered in the discussion focus solely on answering the given questions, which explain the widespread usage of BERT. It is essential to note that BERT was only responsible for question answering, not for question generation — the heart of AQG. Existing systems have been developed surrounding T5 in the provision of questions and answers from passages of text, such as that of [11], where the principles discussed at the beginning of the literature review were applied, along with the implementation of a fine-tuned version of the T5 model. Specific portions of the given input text were the inputs to their model, and a BERT-based model was utilized to evaluate their model. Passages are fed into the model where question and answer pairs are provided.

SQuAD

T5 must be configured to process the appropriate context. The Stanford Question Answering Dataset (SQuAD) is a dataset containing question-answer pairs as well as the passages that were used as the basis [12]. This dataset is commonly used as a tool for the fine-tuning process of T5, which allows for AQG. This enables the given T5 model to perform the correct analysis and procedure for analyzing text context and generate the correct question based on the given keyword. Ultimately, the combination of T5 and SQuAD may enable the generation of question-answer pairs that adhere to the consensus framework in the literature.

Table 2.2: Sample data entry in SQuAD (sourced from [12]**)**

ID	573610e1012e2f140011a182
Title	Hunting
Context	Hunting big game…
Question	What is required of hunting migratory waterfowl?
Answers	{ "text": ["duck stamp"], "answer_start": [439] }

Evaluation of AQG

As defined by Wang et al. in their study on the evaluation of language-processed outputs, much of the review of language-processed outputs, such as summaries and question-answer pairs, revolves around the metric of similarity to a reference text [13]. In other words, testing and validation in this research context are based on comparing accurate basis texts. The F-score is a core metric that describes the tendency of a prediction to be accurate as a function of the precision and recall of the specific classification (or prediction). An extension of this notion is the ROUGE and BLEU metrics. These both come from the principle of finding similarities and context-retention between a generated text and a reference text [13]. The difference between ROUGE and BLEU comes from the focus. ROUGE focuses on context retention, while BLEU focuses on syntax similarity. The researchers behind the AQG systems mentioned in the previous discussion used these metrics as their statistical treatment for testing and validating the respective systems.

Optical character recognition

Once again, there are currently no automatic question generation (AQG) systems that handle handwritten lecture notes, highlighting the need for an optical character recognition

(OCR) framework to enable text extraction from written sources. Existing AQG systems, though effective, are limited to processing digital text and do not support direct extraction from handwritten notes. For instance, [14] incorporated text extraction from digital documents but did not extend this capability to handwritten content. Similarly, a survey by Arbaaeen and Shah categorized AQG systems based on their question-generation methods and information-processing techniques, but found that no systems were designed for handwritten lecture notes [5]. The study further emphasized the need for more advanced AQG systems capable of extracting information from physical media, including handwritten notes. Given these limitations, integrating OCR into AQG systems presents a viable solution for bridging this gap.

Convolutional Neural Networks

Many kinds of neural networks serve a specific purpose. One such neural network relevant to this research endeavor is the convolutional neural network (CNN). As explained by [15], this type of neural network is used in image classification. Building on the groundwork from the previous discussion regarding neural networks, convolutional networks differ from standard multilayer perceptrons by having dedicated layers within their hidden layers. One of these layers is called a convolutional layer, which enables the transformation of a specific image, allowing features to be extracted. These gathered features are simplified with pooling layers that generate sets of values that can be utilized more easily in succeeding computations. Lastly, a connected layer allows for the eventual classification based on the perceived features of the image. On the application side, CNN covers a wide range of use cases, with the majority falling under-recognition and classification tasks. For instance, the study by Sutayco and Caya [27] utilized MobileNetv2 and VGG-16 in identifying a specific kind of mushroom for medical purposes. Their research highlighted how architectural variants of CNNs are well-suited to

particular purposes. Similar to the research involving CNN, their hardware implementation utilized a Raspberry Pi, along with camera interfaces and a custom-built housing, to capture the objects considered in their study. Examples of such systems can be found in the work of Villaverde et al., where fabric types were classified based on the type of cotton used. They utilized a combination of a Raspberry Pi microcomputer and image-capturing tools. Another implementation involved the detection and classification of plant diseases based on image captures of the affected.

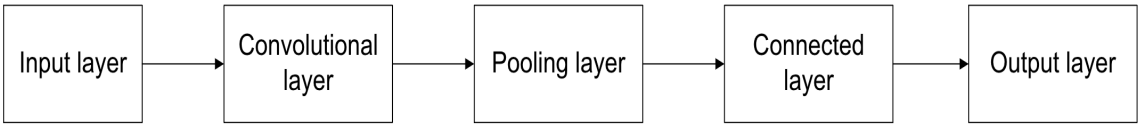


Figure 2.2: A basic representation of a CNN model

Figure 2.2 shows the general CNN flowchart. In simpler terms, the hidden layers of a CNN allow for extracting more specific features from the images that allow for the detection of patterns [18]. OCR utilizes CNN to classify physical text into words that are represented in machine data. This allows physical text originating from documents to be gathered by a system that can be interpreted in specified ways, such as the system proposed in this research endeavor. A specific combination of convolutional, pooling, and fully connected layers may enable a system that efficiently extracts features.

The paradigm for CNN-based handwriting feature extraction also aligns with the frameworks presented in the literature. Kumar and Mishra implemented a system that follows the paradigm where images are subject to some form of image augmentation, allowing for the ease of feature extraction. This system then separates each word into individual characters,

where classification is performed [19]. OCR is, however, not strictly limited to Latin alphanumeric characters, as demonstrated by Ligsay et al. with a system that uses a CNN-based network to extract a local handwriting system from the Philippines, called Baybayin [20]. The metrics involved in their testing and validation centered on using a confusion matrix, which utilized both a test and a validation set. Similarly, Padilla et al. employed an Inception-v3 CNN model to extract features of handwritten Gregg shorthand stenography images into English words [21]. Model predictions were compared to the actual Gregg-to-English translations for legal terms. However, some legal terms in Gregg shorthand writings were incorrectly translated due to minuscule changes in handwriting. Even a signature validation system has been realized using a CNN Inception-ResNet architecture by Ishikawa et al. to eliminate signature forgery cases by analyzing and comparing sample signatures [22].

The CNN-based studies utilized a microcomputer, such as a Raspberry Pi, to host the process and interfaced with peripherals, including a web camera, monitor, keyboard, and mouse [28] [29]. Custom-made external compartments were also utilized, which provided additional lighting to aid the eventual image augmentation of model processing later in their implementation [29]. Moreover, graphical user interfaces (GUIs) integrated into the models were displayed on monitors to guide the user in capturing images, allowing them to use other software functions or save the model results [21] [22].

TrOCR

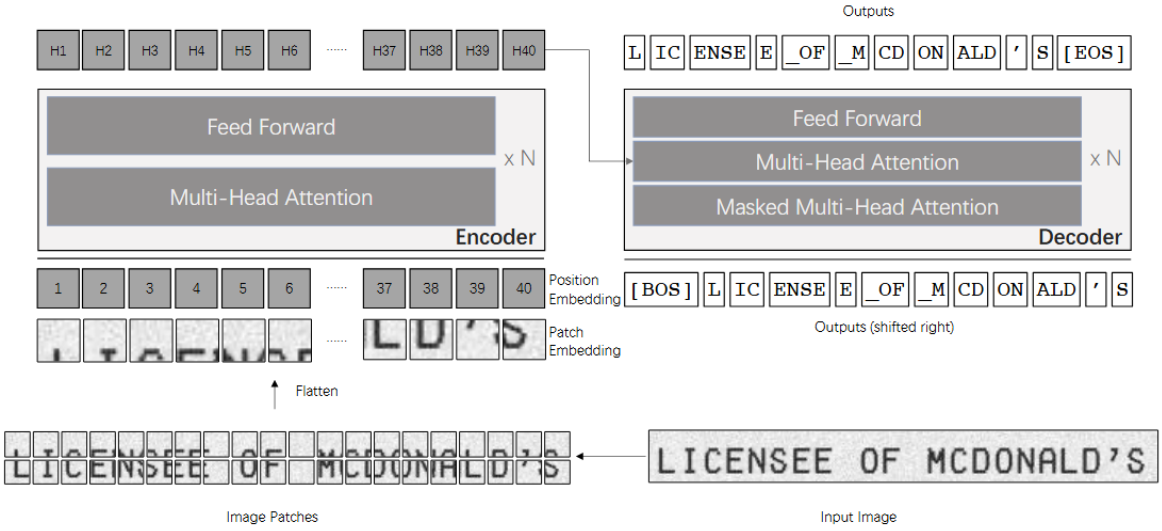


Figure 2.3: Model architecture of TrOCR (sourced from [23]**)**

Eventually, the widespread use of CNN in its various forms led to the development of advanced network architectures, such as the transformer. A prime example of this is the TrOCR model. It is an open-source transformer model that can be fine-tuned for a specific dataset. The use case of this transformer model specializes in extracting text from digital or physical forms [24]. Other interesting use cases of TrOCR also come from specialized handwritten text extraction, such as that of Arabic text in the study by Mortadi et al. [25]. The use of transformers, as opposed to CNN architectures, stems from the fact that CNN architectures operate on a global level. In contrast, the utilization of a transformer architecture made possible through the addition of a recurrent neural network (RNN) allowed for a much deeper analysis and context-awareness in the model's processing [23]. The outcome of an OCR transformer model, such as TrOCR, tends to be better than traditional OCR architectures built on CNN.

Evaluation of OCR

OCR is a task that predicts accurate handwritten text. As such, the degree of truth concerns some form of basis. Much of the existing OCR implementations take on a variety of evaluation methods. Starting with the first metric, a confusion matrix is used to describe the performance and accuracy of a model's classification task. It is also a graphical tool for visualizing the distribution of accuracy and error in a model's classification, with the diagonal elements representing correct classifications and the off-diagonal elements representing inconsistencies [26]. For instance, a use case of identifying expiry text from canned goods by Manlises et al exemplified the use of a confusion matrix [27]. The use of confusion matrices is not confined to OCR. Studies, such as that of Padilla et al., where they utilized computer vision in a classification task on mollusk species [28]. Similarly, Padilla et al. enabled the development of a checkout system utilizing computer vision and classification, with evaluation conducted using confusion matrices [29]. However, use cases such as extracting and classifying words from existing vocabularies cannot be described by confusion matrices. A study led to the implementation of a system that extracted and categorized handwritten Myanmar characters [30]. Their evaluation took the form of analyzing patterns of similarity and thought capture through various metrics. In the capture of the degree of similarity concerning a ground truth basis, the F-score lays a mathematical model of such metric. As highlighted in the text extraction use case of Zhang et al, the metrics involved centered around the usage of the F-score [31]. Coming from the F-score, other studies utilize metrics that are derived from it such as the word error rate (WER) which as the names suggest, allowed for the description of the degree of performance of a model tasked in extracting text which are highlighted in similar use cases under [28] [32].

Chapter 3

METHODOLOGY

General Methodology

For the study, an experimental research design will be utilized. The research design's premise revolves around testing the effectiveness of the AQG system in generating questions from passages of physical handwritten lecture notes using TrOCR and the T5 large language model (LLM). A prototype is constructed to facilitate OCR and contains the necessary hardware components: a Raspberry Pi 5, a web camera, and a touchscreen display. For prototype testing, the researchers collected handwritten notes from undergraduate students. The system is then evaluated using WER and WOER metrics for the OCR and ROUGE and BLEU metrics for the AQG operation.

Conceptual Framework

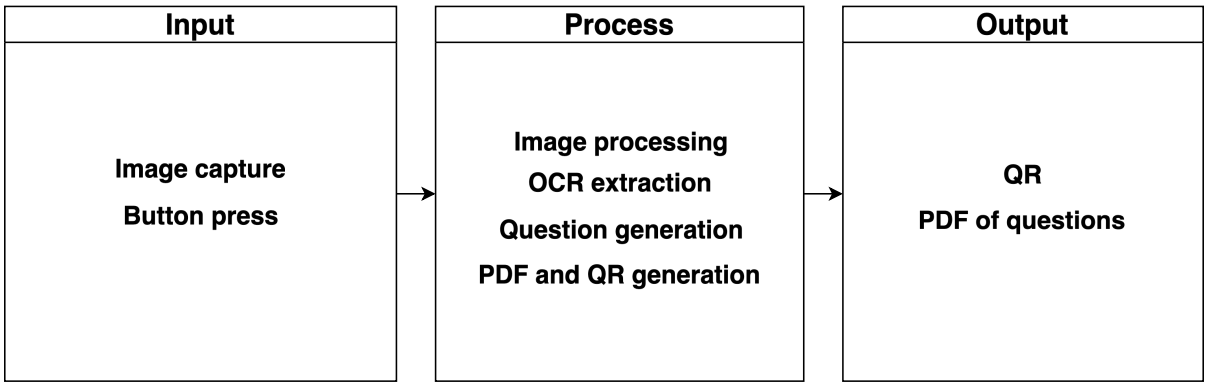


Figure 3.1: Input-Process-Output framework

Figure 3.1 illustrates the conceptual framework of the system. The system expects user button presses on the interface to capture the notes. Upon receipt of the images, a series of

image preprocessing steps, including denoising, dilation, thresholding, and binarization, will be performed. Contour detection will facilitate text line extraction through preprocessing, which occurs before inference. Batched TrOCR inferencing is done on the collection of text lines that will produce text data that will be sent to Gemini for text enhancement and correction. Afterwards, relevant keywords are extracted using spaCy, and Rapid Automatic Keyword Extraction (RAKE) will be done, which creates a set of appropriate words and phrases. T5 inferencing utilizes relevant words and phrases alongside the enhanced text to generate questions. A PDF containing the formatted questions will be uploaded to Google Drive via its API, allowing for the creation of a link that can be transformed into a QR code for presentation.

Hardware Development

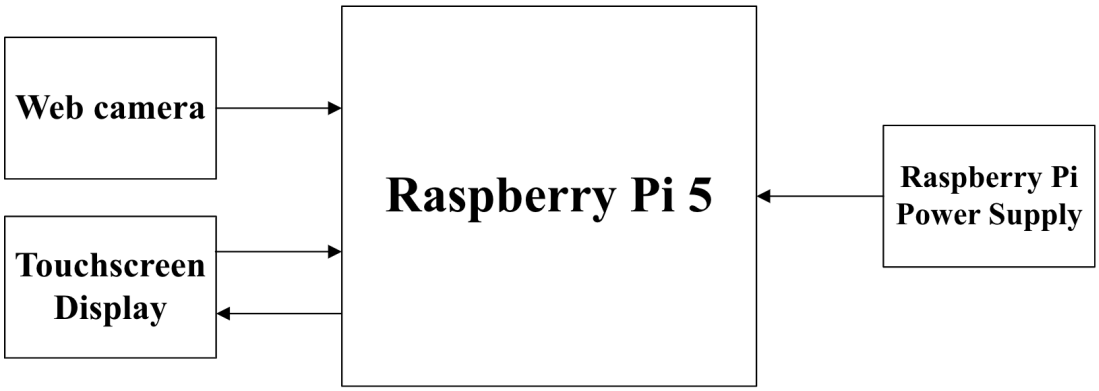


Figure 3.2: Hardware organization block diagram

Figure 3.2 is the hardware organization of the system in the form of a block diagram, and the associated interfaces utilized for the system. All in all, these are the components that will be used for the hardware development of the device: a Raspberry Pi 5, a microSD card (micro standard digital), a USB web camera, an LCD touchscreen monitor, and a power supply.

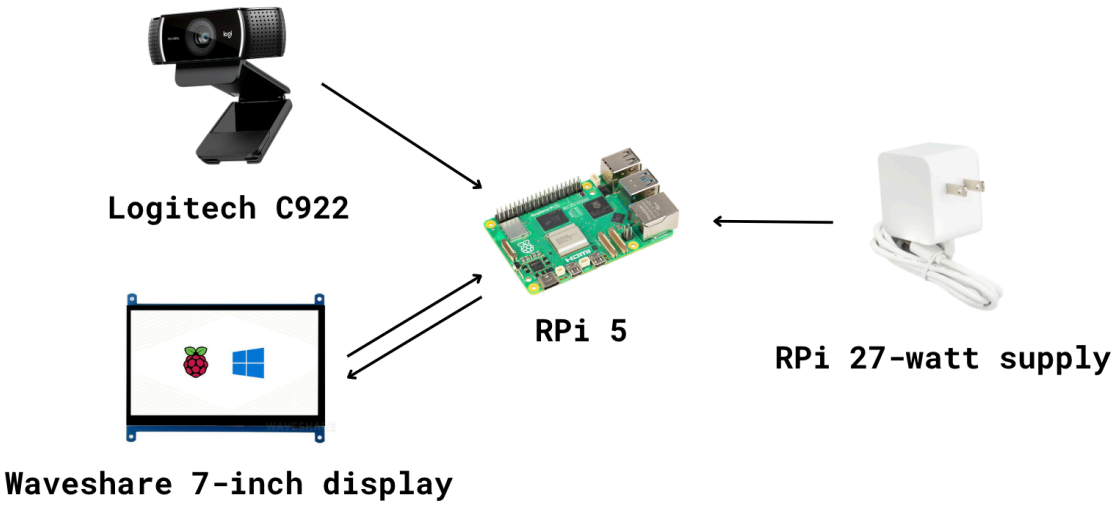


Figure 3.3: Graphic hardware diagram

Shown in Figure 3.3 is the proposed hardware implementation with the respective devices. The Raspberry Pi 5 will be the processing unit for the software implementation and algorithms in this system. The Logitech C922 web camera will take 1920x1080 captures of the two halves of the handwritten notes. The microSD card enables the storage of the system's outputs. The touchscreen monitor is used for the system's graphical user interface (GUI), which handles user inputs and system outputs during operation. Finally, a power supply provides power to the Raspberry Pi 5 and other hardware components of the system as it is interfaced with the different hardware components, such as the USB web camera that the OCR input pipeline will use, a microSD card that allows for the storage of the outputs of the system, and a touch screen monitor that provides for the inputs to be given by the user and outputs in the

form of statuses of the system during the operation of the system. Lastly, a power supply will be provided to the system for operation.

Software Development

A. Software architecture

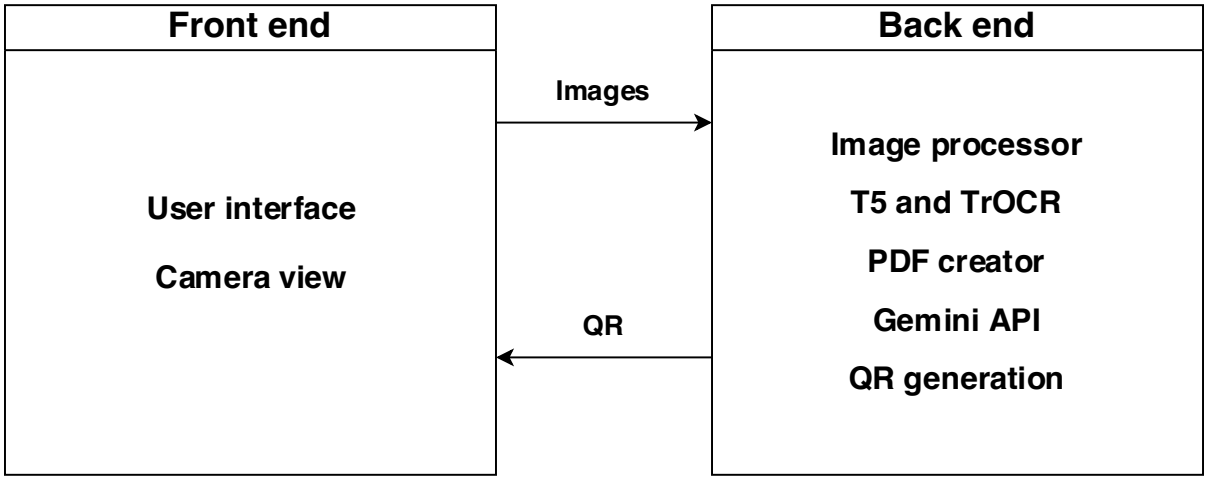


Figure 3.4: Software program architecture

As shown in Figure 3.4, the software for this system comprises both the front-end and the back-end. Starting with the front end, it contains the user interface that utilizes the Angular framework. The user interface features a camera feed and buttons that enable the user to capture the given notes. After confirmation of capture, the front end sends the images in Hypertext Transfer Protocol (HTTP) payloads to the back end for processing. In terms of the back end, it uses the Flask framework. It contains the image processing pipeline, which utilizes OpenCV tools and will be discussed further in the chapter. Moreover, the T5 and TrOCR models are loaded in the program memory for OCR and AQG. To create the questions, a PDF containing the questions will be generated using ReportLab. The generated PDF will then be uploaded to Google Drive through its API, where the download link is presented in the form of a QR code through the front end.

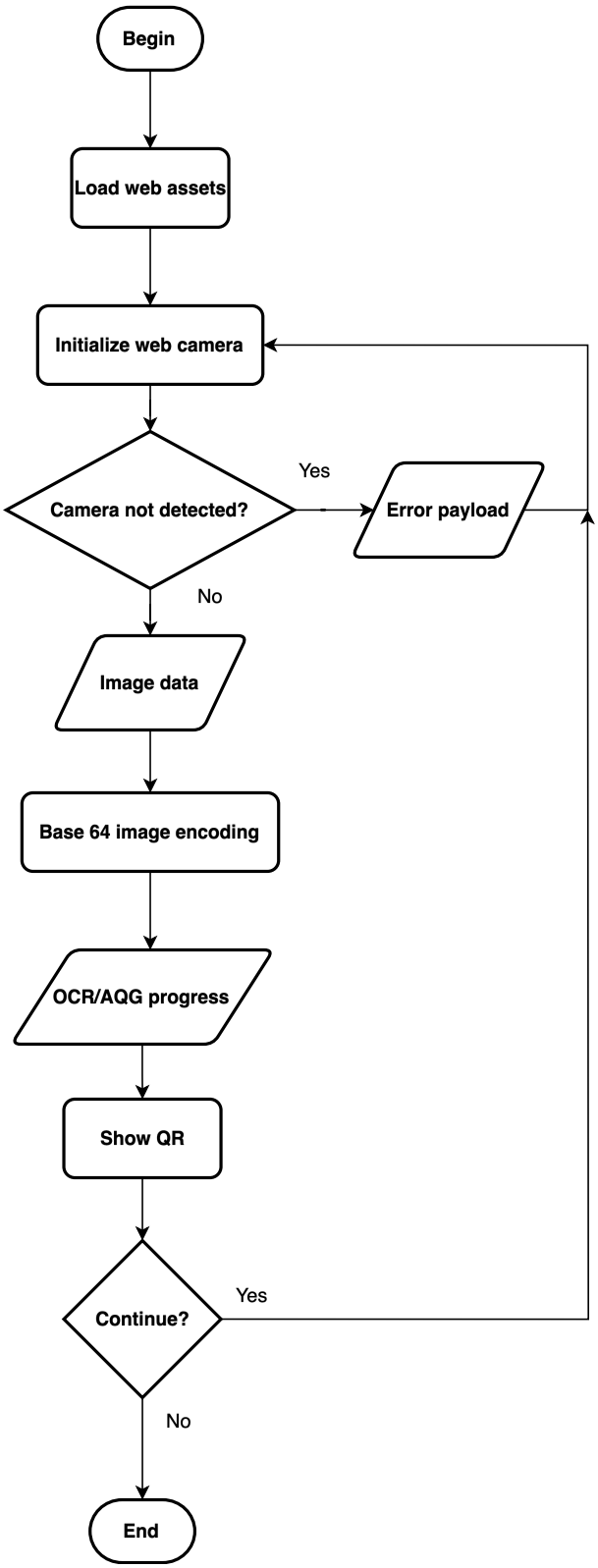


Figure 3.5: Front-end flow.

Figure 3.5 shows the flowchart for the front-end program flow. Since the front end is served as a web application, it first loads the web assets from its source directory. Afterwards, the system's web camera is initialized in the interface. If the camera does not work, the program releases an error payload for debugging and then re-attempts to initialize the camera. Once the camera is operational, the user captures images, which allows the front end to send the photos to the back end. Progress is continuously monitored through repeated HTTP requests, displayed as a progress bar on the user interface. After processing, the QR code for the PDF download is presented, where the user can expect to find the questions within the PDF file. Should the user want to continue, the process repeats with the initialization of the camera.

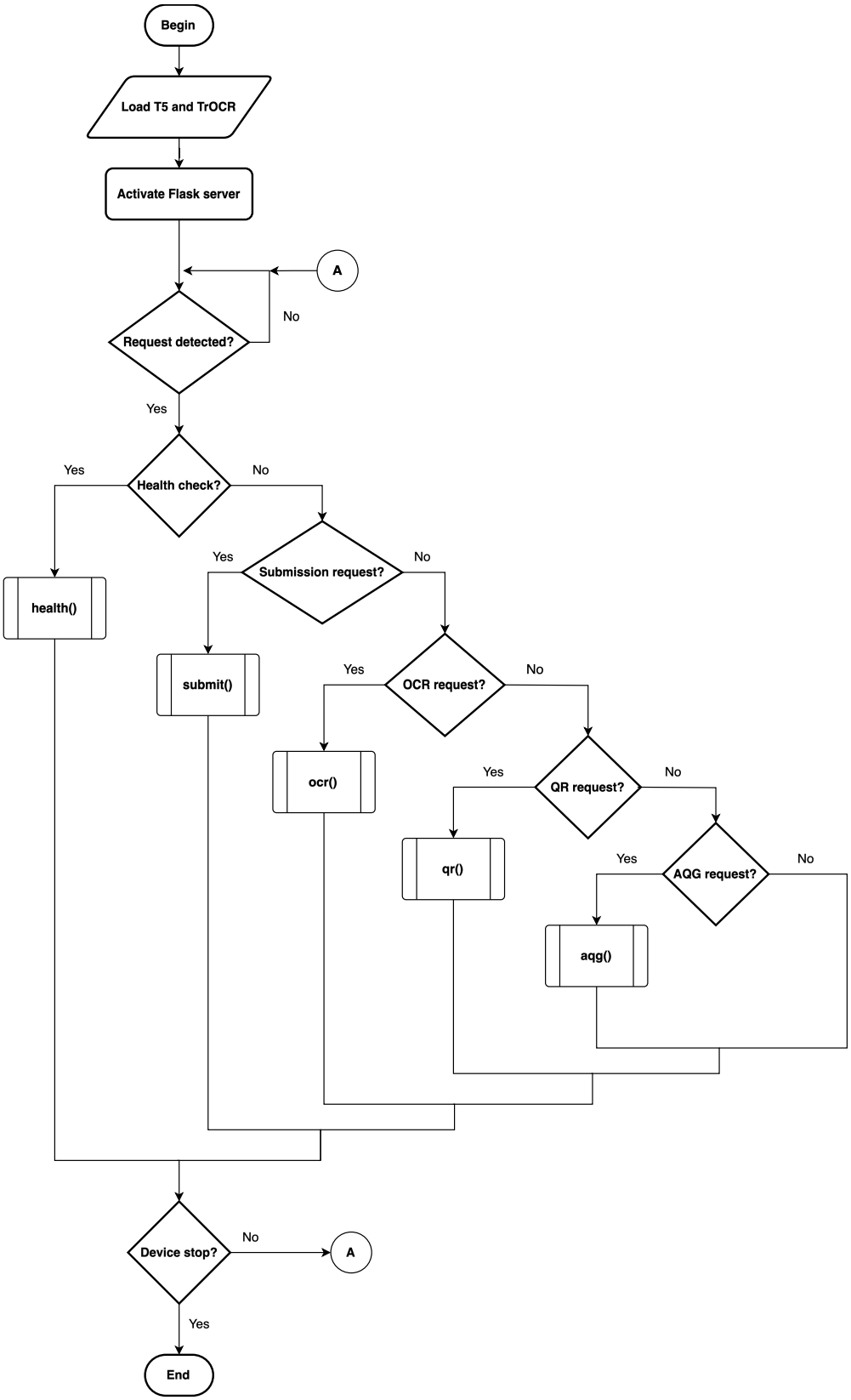


Figure 3.6: Back-end front-end flow

Figure 3.6 is the program flow of the back end. In the initialization, the program loads the T5 and TrOCR models from the source directory. Afterwards, the Flask server is run with the associated API helper functions defined by the series of conditional statements. There are four functions within the back-end program: the `health()`, `submit()`, `ocr()`, `qr()`, and `aqg()` functions. Depending on the kind of request the back end receives from the front end, it invokes the corresponding function. If no requests are detected, it repeats the request checking. The program terminates when the web application or device shuts down.

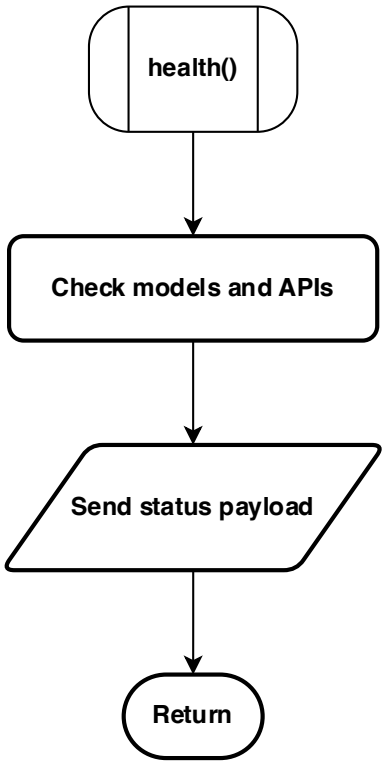


Figure 3.7: Health check subroutine flowchart

Figure 3.7 defines the program flow for the health subroutine, which checks the consistency of models and API calls within the back end. In the event of an error, the subroutine outputs a status payload that alerts the front end to an error.

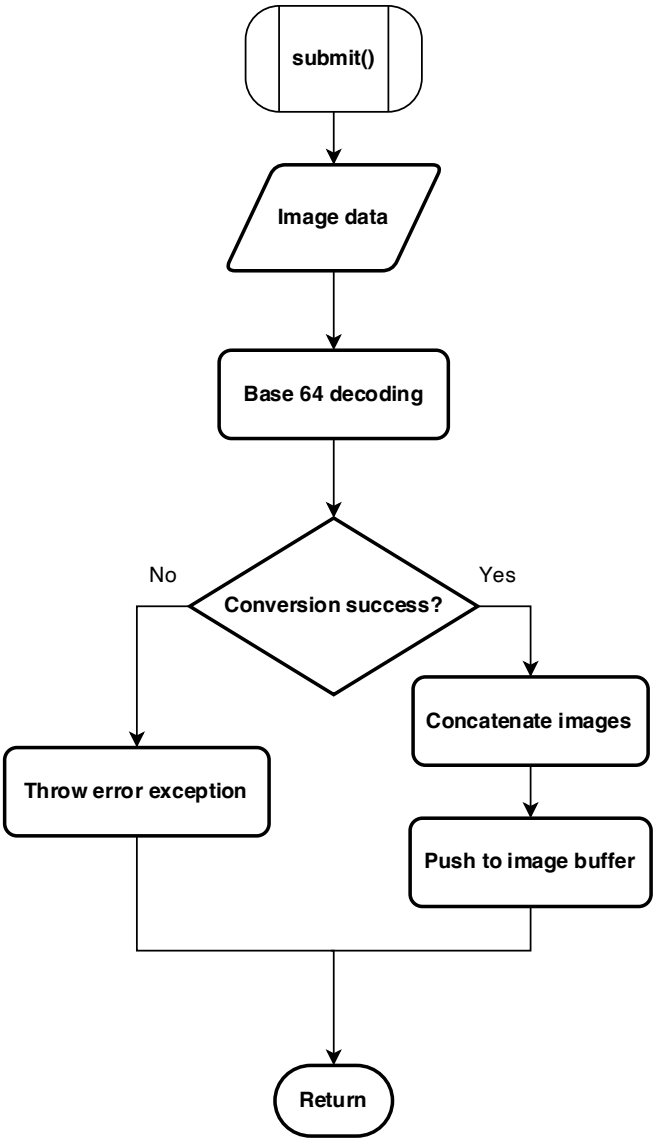


Figure 3.8: Submit subroutine flowchart

Figure 3.7 explains the submit subroutine if the back end is to receive inbound image captures of the user's notes. Image data is obtained from the HTTP payload, where base-64 decoding transforms the image base-64 strings to OpenCV image objects. If the transformation is successful, the two halves of the pictures are concatenated as a single image that is pushed to the image buffer for processing. Conversely, an unsuccessful image conversion leads to an exception.

Figure 3.9: OCR subroutine

Figure 3.9 shows the program flow for the OCR subroutine of the system. The subroutine begins with the decision whether the image buffer is nonempty. Otherwise, the subroutine throws an error. The subroutine operation works by iterating over the images in the image buffer. For each of the photos, another subroutine called `extract_lines()` returns an array of OpenCV image objects that contain the individual text lines in the document. Another loop occurs where each text line is encoded and decoded through TrOCR, which extracts the text. All the extracted text is concatenated into a single string and enhanced through Gemini, which is set as a global variable.

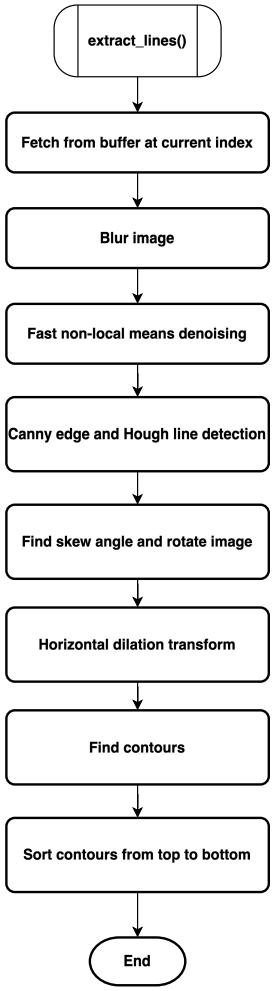


Figure 3.10: Line extraction subroutine

Figure 3.10 explains the text extraction algorithm in the OCR subroutine. In the given iteration on the image buffer, the image is subject to Gaussian blurring and fast non-local means denoising to eliminate blemishes on the image capture. Canny edge and Hough transformations reveal the skew angle of the text lines for optimal horizontal alignment. Afterwards, horizontal dilation occurs with a kernel of 300 by 150 pixels, showing the individual text lines as rectangles. Contour extraction follows, allowing for the natural sequencing of text lines from top to bottom through a sorting process. These contours serve as a guide for cropping the original image to extract the text lines.

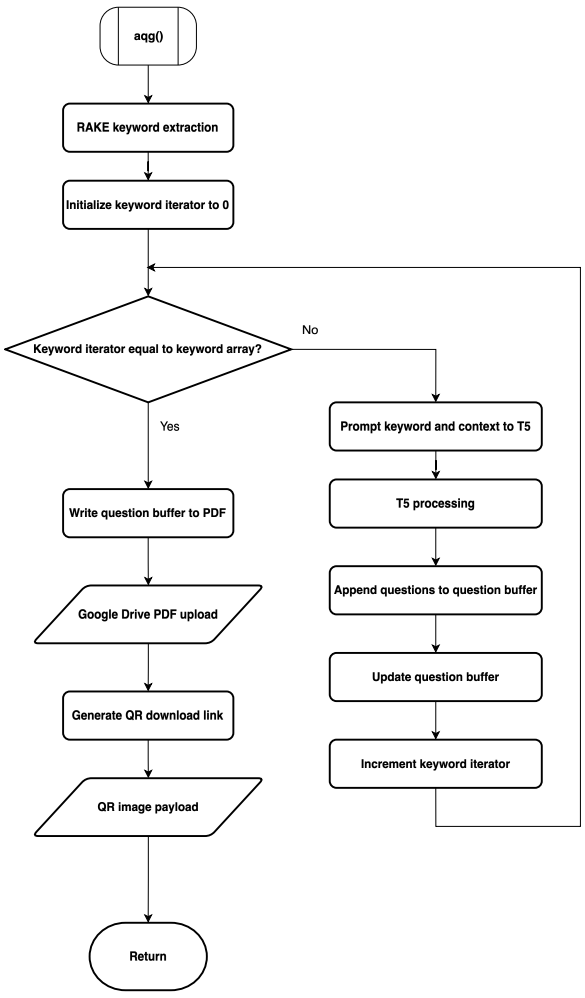


Figure 3.11: AQG subroutine flowchart

Figure 3.11 is the program flow of the AQG subroutine. RAKE keyword extraction occurs on the enhanced text from the OCR process, resulting in an array of relevant keywords and phrases. Each keyword is iterated over, prompting T5 for question generation with each iteration. The generations are appended to the question buffer of the back-end program. After the question generation, a PDF is created from the generated questions and is consequently uploaded to Google Drive via its API. The download link for the PDF is converted into a QR code, which is sent to the front end for presentation.

B. Model Training

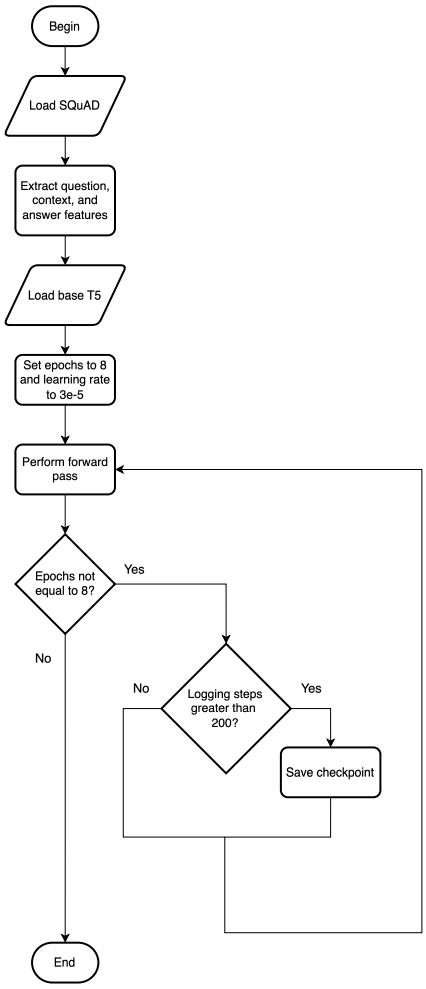


Figure 3.12: T5 fine-tuning flowchart

Figure 3.12 shows the fine-tuning procedure for T5. Using SQuADv1.1, T5 was trained to receive a paragraph and a chosen keyword and output a corresponding question. In doing so, the answer, question, and context (paragraph) columns were extracted from SQuAD. The base version of T5 was loaded and trained for eight epochs at a learning rate of 3×10^{-5}. Logging was implemented, where a checkpoint was saved every 200 steps for backup.

Experimental Set-up

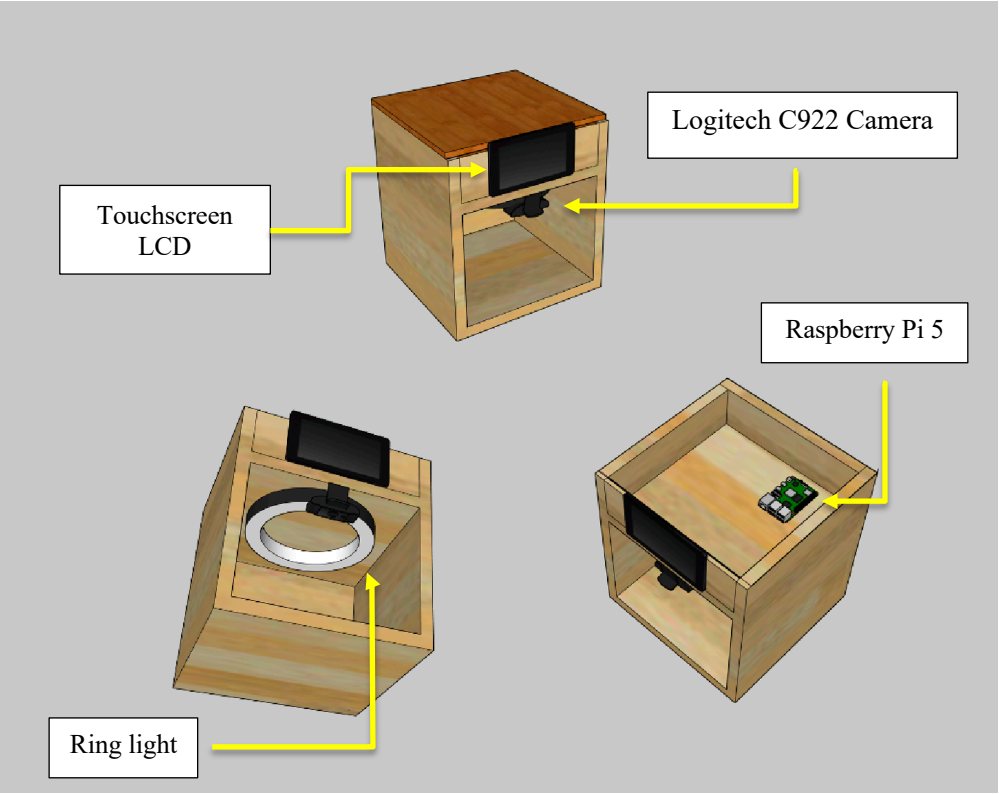


Figure 3.13: Experimental setup

Figure 3.13 presents the experimental setup in a render depiction. Here, the proposed components are interconnected within an enclosure made of 2 cm-thick plywood with a one cm-thick wooden lid. In the experimental setup, a ring light will also be supplied for

illumination. In the experimental usage, users will be expected to place one page at a time within the enclosure, as will be visible through the camera feed on the screen.

Data Gathering

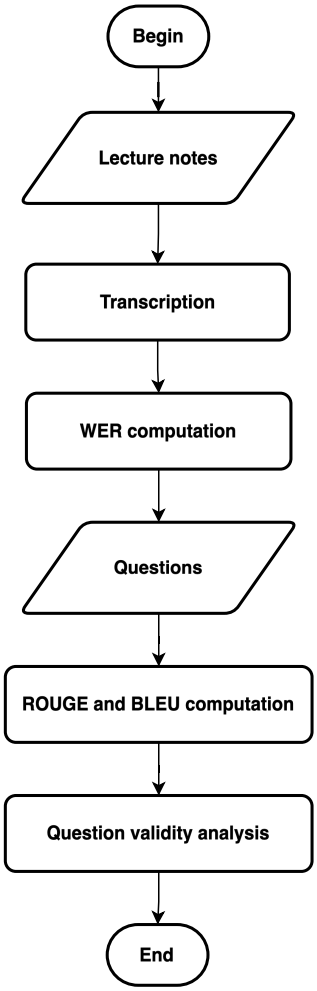


Figure 3.14: Data gathering procedure flowchart

Figure 3.14 explains the data gathering procedure of this research. The purpose of data gathering is to test and evaluate the system. Data collection will begin with the collection of 70 handwritten notes from students enrolled in undergraduate cybersecurity courses. The handwritten text will be manually transcribed and compared with the TrOCR extraction

through the computation of the Word Error Rate (WER) per text line. Afterwards, 135 questions will be collected from students, and they will be compared with the model-generated questions using ROUGE and BLEU.

Table 3.1: Testing table for OCR

Index	**Handwritten Text**	
	Student writing *(sample)*	**Model extraction** *(sample)*
1	Establishing a robust operating system security baseline is crucial for protecting systems from a multitude of threats. This process begins with identifying potential vulnerabilities and establishing a secure baseline configuration. It's vital to remember that even seemingly harmless scripts can be malware, and users should be wary of rogue antivirus products that can compromise security. Fileless attacks present a significant challenge as they are difficult to detect and remove, underscoring the need to always remove unapproved software.	Estblishing a robost operatng systm security basline is cruial for prtecting systms frm a multtude of threts. This proess begins with identifyng potental vulnerabilitis and establishng a secure basline configration. It's vita1 to remmber that even semingly harmlss scipts can be malwae, and users shuld be wary of rogue antivrus products that cn compromse security. Fileles atacks presnt a signifcant chalenge as they are dificult to detct and remove, underscoring the ned to alwys remove unaproved softwae.
…	…	…
70	Malware detection employs various techniques, including signature-based methods that recognize characteristics of known malware and heuristics-based approaches that identify general features shared across malware types. Behavior-based detection further enhances this by analyzing suspicious behavior. For endpoint security, a host-based firewall and comprehensive host-based security suites are essential.	Malwaer detectin employs variuos techniqes, includnig signatrure-based recogizes various charcateristics of knwn malware anmd heurstics-based recogrnizes genral features shard by variuos types of malwaer. Behavir-based emplys anlysis of suspicous behvior. For endpont secrity, a host-basd firewal and host-bassed security suies are essntial. Netwrk-based malwae protction and advancd malwae protection end piont protectin frm virses and malwae. Netwrk admision conrol (NAC) permts anly authoized and complint systms. Hos basd intrustion preventin, cabl whil swtch pc is conected via straght thrughby manualy adjustng the 1P addres to.

Defined in Table 3.1 is the testing table for Optical Character Recognition (OCR). The 70 handwritten notes collected from the students will be manually transcribed and placed in this table alongside the TrOCR extracted text. Moreover, each handwritten lecture page will generate five questions by the system, as shown in the table below.

Table 3.2: Testing table for question validity

Question index	Question *(sample)*
1	What works alongside the Ethernet?
2	What role does MAC fulfill in networking?
…	…
349	In networking, what do you call to the address that help manage data transmission and network access?
350	In networking, which has a crucial role for managing device identification and communication?

Table 3.2 presents the testing table for the validation of question. Each handwritten note (as shown in the note index) will correspond to five generated questions as indexed by the question index. In total, there will be 350 questions from the intake of 70 handwritten notes.

Table 3.3: Testing table for question comparison .

Index	Question comparison	
	Student *(sample)*	**Model** *(sample)*
1	What works alongside MAC address to ensure seamless data transfer between connected devices.	What does MAC work alongside to ensure seamless data transfer between connected devices?
2	What is the traditional technology for connecting devices in a wired local area network (LAN) or wide area network?	Along with MAC, what other network allows seamless data transfer between connected devices?
…	…	…
135	What works alongside MAC to ensure seamless data transfer between connected device?	What MAC interfaces with MAC to ensure seamless data transfer between connected devices?

On the other hand, Table 3.3 is the testing table for evaluating the similarity of questions to student-made questions. Here, the 135 questions collected from the students are accumulated in this table. Each question will be compared to its corresponding model-generated question.

Statistical Treatment

Utilizing the testing tables from data collection, the following evaluation tables will discuss the statistical method to be used.

Table 3.4: Evaluation table for OCR

Index	**WER** *(sample)*
1	0.23
…	…
70	0.34
Average	**0.33**

Table 3.4 shows the corresponding WER value across the entries in Table 3.1 for the extracted text. The WER is computed as follows: where cap S i.Sit is the number of total words in the generated questions.

Table 3.5: Evaluation table for question validity.

Note index	**Question index**	**Valid?**
1	1	Yes
	2	Yes
	3	No
	4	Yes
	5	Yes
…	…	…
70	346	Yes
	347	Yes
	348	Yes
	349	Yes
	350	No
Total valid		**329**

Table 3.5 presents the evaluation table for the question validity of the questions generated from the handwritten notes. Here, the professor of the cybersecurity course manually evaluated the studen' questions to determine whether they correlated with the answers and if the questions were correct in terms of grammar and syntax. The question validity rate is computed as follows:

$$\text{Question Validity} = \frac{\text{valid questions}}{\text{total questions}} \times 100\% \quad (2)$$

Table 3.6: Evaluation table for AQG

Index	ROUGE	BLEU
1	0.492	0.336
…	…	…
70	0.766	0.512
Average	**0.442**	**0.379**

On the other hand, Table 3.6 is the evaluation table for the model-generated questions using the ROUGE and BLEU metrics. ROUGE is computed as follows:

$$\text{ROUGE} = \frac{1}{N}\sum_{i=1}^{N}\frac{2 \times P \times R}{P + R} \tag{3}$$

Where P is the precision, R is the recall, and N it is the number of words in the question. P and R are computed as follows:

$$P = \frac{\text{number of n-gram overlap}}{\text{Total n-grams in generated question}} \tag{4}$$

$$R = \frac{\text{number of n-gram overlap}}{\text{Total n-grams in reference question}} \tag{5}$$

As for the BLEU metric,

$$\text{BLEU} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{r}\sum_{i=1}^{r}\min\sum_{i=1}^{r}\text{count}_{\text{clip}}(ngram) \tag{6}$$

where N is the type of n-gram, r is the length of the reference sequence, and $\text{count}_{\text{clip}}(ngram)$ is the clipped count of n-grams.

Chapter 4

RESULTS AND DISCUSSION

Through the application of testing and evaluation, the researchers were able to collect 70 handwritten notes and perform evaluations for OCR and AQG.

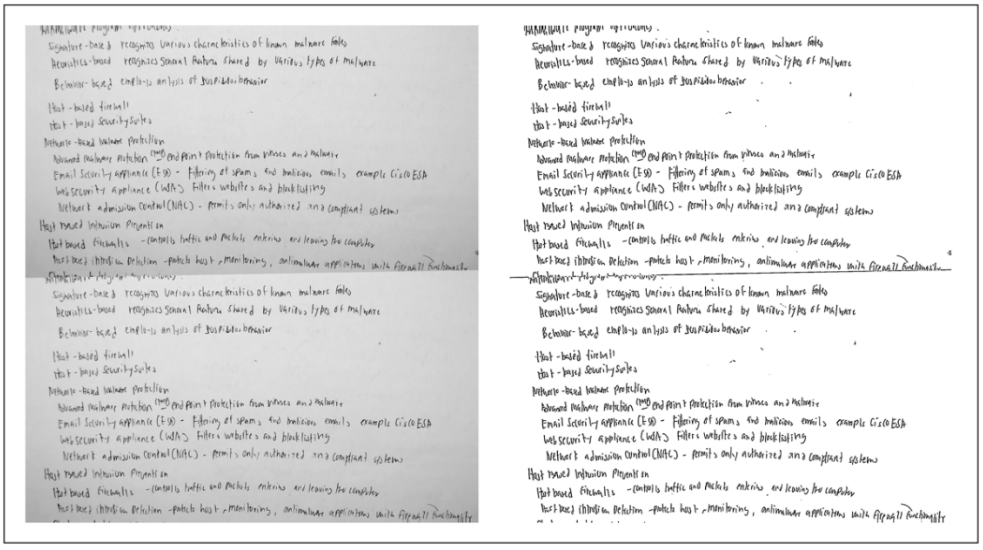


Figure 3.15: Sample handwritten note capture (left) and processed version (right)

Figure 3.15 shows an example capture of a handwritten note within the testing set with a corresponding processed version using the methods shown in the software development. The 70 handwritten notes were captured and documented within the source directory of the Raspberry Pi 5, alongside a tabulation using a comma-separated value (CSV) format, following the testing and evaluation table formats.

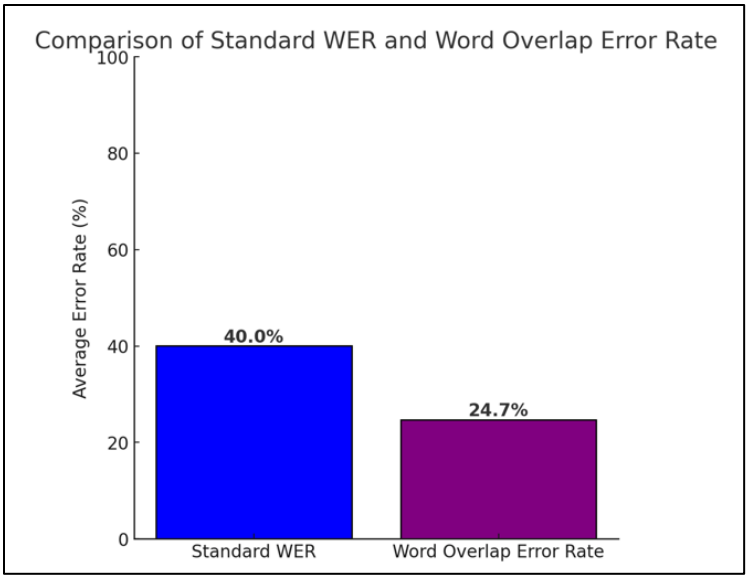


Figure 3.16: WER and word overlap error rate (WOER) histogram

Figure 3.16 presents a histogram of the 40% WER and 24.7% WOER averages achieved by the system. The magnitude of the WER was believed to come from three aspects. First, the system found difficulty with the natural horizontal misalignment of the handwritten text. Second, the T5 and TrOCR models would have required more parameters, specifically in the form of large or extra-large (XL) models. Third, the capture resolution of the Logitech C922 web camera may have affected the text extraction. Nonetheless, the 60% word correction rate still gave the Gemini model enough room for the correction of errors, as seen in the following discussion.

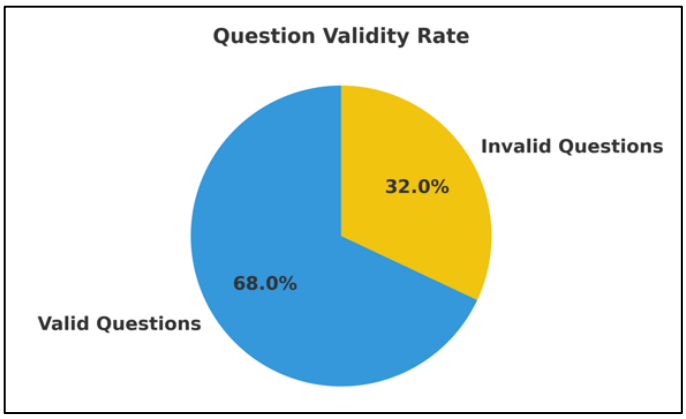


Figure 3.17: Question validity pie chart

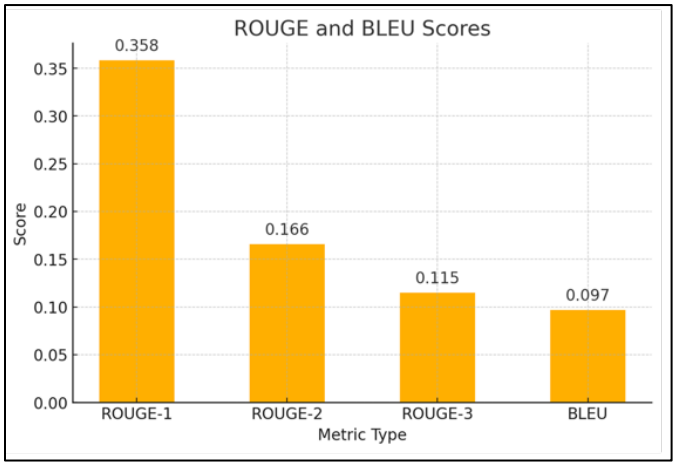


Figure 3.18: ROUGE and BLEU histogram

Figures 3.17 and 3.18 show the question validity and the ROUGE and BLEU scores for the evaluation of AQG. Starting first with the question validity, 238 out of the 350 questions generated from the 70 handwritten notes were in proper grammar, syntax, and correlation to the answer. It was realized that much of the invalidity was caused by missing parts of speech and incorrect grammar, which was believed to be due to the small parameter count of the T5 base model. Moreover, there were questions that immediately reveal the answer within the question. Some questions utilize a WH type that does not correspond to the answer. In terms of the ROUGE and BLEU metrics, the questions decently capture the word-for-word context

from the questions made by students, as indicated by a unigram ROUGE score of 0.358. However, the system encountered difficulty with phrasal similarity, as indicated by the bigram (ROUGE-2) and trigram (ROUGE-3) metrics of 0.166 and 0.115, respectively. Moreover, the BLEU score of 0.097 indicates that the model-generated questions differ significantly from the structure of student questions. Upon analysis, the students tended to use the structure and exact wordings from the reference text. On the other hand, the model-generated questions had the tendency to utilize the relationships within the other text found in the paragraph.

Chapter 5

ENGINEERING STANDARDS

The hardware components utilized in this study adhere to several international standards. This chapter provides a detailed discussion of the devices and their compliance with relevant international standards.

Raspberry Pi 5

The Raspberry Pi 5 has been verified to comply with global regulatory frameworks, notably satisfying the stipulations of the Radio Equipment Directive (2014/53/EU) to affirm its safety and operational integrity. This compliance is achieved by adhering to a set of recognized and accepted standards.

For its radio frequency (RF) functionalities, the device's performance in the 2.4 GHz spectrum is governed by the guidelines for wideband data transmission found in ETSI EN 300 328 V2.2.2 (2019). Similarly, its operations within the 5 GHz range are aligned with the criteria set forth in ETSI EN 301 893 V2.1.1 (2017) and the draft standard ETSI DRAFT EN 301 893 V2.1.51 (2023).

To ensure electromagnetic compatibility (EMC), the Raspberry Pi 5 meets the requirements for radio equipment and services as detailed in ETSI EN 301 489-1 V2.2.3 (2019) and ETSI EN 301 489-17 V3.2.4 (2020). Furthermore, its design and construction adhere to the electrical safety benchmarks established in EN IEC 62368-1:2020+A11:2020 and EN 62311: 2008.

Specifically, the ETSI EN 300 328 v2.2.2 standard ensures that the device's transmissions are confined to the 2400 MHz to 2483.5 MHz frequency range, in accordance with Clause 4.3.2.2 for equipment utilizing wideband modulation. Concurrently, adherence to ETSI EN 301 893 v2.1.1 confirms its operation within the designated 5 GHz bands of 5,150–5,250 MHz, 5,250–5,350 MHz, and 5,470–5,725 MHz. These comprehensive measures underscore the Raspberry Pi 5's conformity with essential standards for safe RF output, electromagnetic compatibility, and user safety. Logitech C922 Web Camera

The Logitech C922 was designed to meet the standards of electromagnetic compatibility (EMC), Universal Serial Bus (USB) connectivity, and radio frequency (RF) interference. Starting with EMC, the device adheres to the standards of multimedia equipment (EN 55032:2015+A11:2020, EN 55035:2017+A11:2020, FCC Part 15 Subpart B), which prevent interference with and from other electromagnetic sources. For USB connectivity, it complies with the electrical safety and human exposure limits for common devices (IEC 62368-1:2018, EN 62311:2008). Regarding RF emission standards, the C922 web camera complies with FCC Part 15 Subpart C and EN 300 328 V2.2.2: 2019, ensuring safe usage in wireless environments.

Waveshare Touchscreen LCD

The Waveshare 7-inch capacitive touch screen LCD complies with the standards of EMC, Federal Communications Compliance (FCC), and electrical safety. First, the device complies with EN 55032:2015+A11:2020 and EN 55035:2017+A11:2020, which pertain to the standard EMC requirements for multimedia devices and the immunity to interference from other devices. Second, the device adheres to FCC Part 15, which sets the limit on electromagnetic interference from digital devices, thereby ensuring electrical safety.

Chapter 6

CONCLUSION

It was realized that the AQG from handwritten notes through TrOCR was possible with a 68% question validity and 40% word error rate. The questions generated approximately have four matching keywords from student-made questions, as shown by the computed ROUGE-1 to ROUGE-3 scores of 0.351, 0.166, and 0.115, and a BLEU score of 0.097, indicating little similarity since the values are closer to 0. It is recommended that further preprocessing of the image captures be performed to improve the text extraction. Additionally, the use of larger versions of TrOCR and T5 is recommended, along with the utilization of more powerful hardware to enhance system processing capabilities and an improved camera for higher image resolutions. It is also recommended that alternative evaluation methods be utilized for this AQG implementation, such as similarity or semantic analyses.

Chapter 7

RECOMMENDATION

To further enhance the accuracy and efficiency of text extraction, further preprocessing of image captures of the lecture notes is strongly recommended. Additionally, utilizing larger and more advanced versions of the TrOCR and T5 models is recommended to enhance the system's performance. To support these improvements, the researchers suggest using more powerful hardware with increased processing capabilities and upgrading the camera to capture higher-resolution images. Furthermore, the researchers recommend exploring alternative methods for text enhancements that may improve the validity and coherence of the system's output questions.

REFERENCES

[1] A. Arbaaeen and A. Shah, "Natural language processing based question answering techniques: A survey," in *7th IEEE International Conference on Engineering Technologies and Applied Sciences, ICETAS 2020*. Institute of Electrical and Electronics Engineers Inc., 12 2020.

[2] D. A. Padilla, A. J. M. Mesina, and E. J. A. Perez, "English treebank of Mapua University smart interactive voice response system," in' *2020 IEEE 12th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*. IEEE, 12 2020, pp. 1–6.

[3] D. A. Padilla, N. K. U. Vitug, and J. B. S. Marquez, "Deep learning approach in Gregg Shorthand word to English word conversion," in *2020 IEEE 5th International Conference on Image, Vision and Computing (ICIVC)*. IEEE, 7 2020, pp. 204–210.

[4] A. N. Yumang, M. J. F. D. Cru, and M. C. Q. Jasmin, "Relationship strength of handwriting consistency and number of local features with the accuracy of a handwritten forgery detection model," in *2022 IEEE 14th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management, HNICEM 2022*. Institute of Electrical and Electronics Engineers Inc., 2022.

[5] Y. Y. Ou, S. W. Chuang, W. C. Wang, and J. F. Wang, "Automatic multimedia-based question-answer pairs generation in the computer-assisted healthy education system," in *2022 10th International Conference on Orange Technology, ICOT 2022*. Institute of Electrical and Electronics Engineers Inc., 2022.

[6] M. Moron, J. Scocozza, L. Chiruzzo, and A. Rosa, "A tool for automatic question generation for teaching English to beginner students," in *Proceedings - International Conference of the Chilean Computer Science Society, SCCC*, vol. 2021-November. IEEE Computer Society, 2021.

[7] H. Gaur, D. K. Tayal, and A. Jain, "Viqg: Web tool for automatic question generation from code for viva preparation," in *Proceedings of 2023 26th Conference of the Oriental COCOSDA International Committee for the Co-ordination and Standardization of Speech Databases and Assessment Techniques, O-COCOSDA 2023*. Institute of Electrical and Electronics Engineers Inc., 2023.

[8] D. C. Tsai, A. Y. Huang, O. H. Lu, and S. J. Yang, "Automatic question generation for repeated testing to improve student learning outcome," in *Proceedings - IEEE 21st International Conference on Advanced Learning Technologies, ICALT 2021*. Institute of Electrical and Electronics Engineers Inc., 7 2021, pp. 339–341.

[9] A. M. P. Ligsay, J. B. Rivera, and J. F. Villaverde, "Optical character recognition of baybayin writing system using YOLOv3 algorithm," in *2022 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*. IEEE, 9 2022, pp. 1–5.

[10] M. J. Y. Sutayco and M. V. C. Caya, "A comparative study of mobilenetv2 and vgg-16 convolutional neural network architectures for identification of medicinal mushrooms." Institute of Electrical and Electronics Engineers (IEEE), 7 2024, pp. 1–6.

[11] C. O. Manlises, D. A. Padilla, J. B. Santos, and P. A. Adviento, "Expiration identification of canned goods using a convolutional neural network," in *2024 7th International Conference on Information and Computer Technologies (ICICT)*. IEEE, 3 2024, pp. 173–177.

[12] A. D. R. Calimag, D. A. Padilla, and C. O. Manlises, "Checkout system with object detection using Nvidia Jetson Nano and Raspberry Pi," in *2023 IEEE 5th Eurasia Conference on IOT, Communication and Engineering (ECICE)*. IEEE, 10 2023, pp. 168–171.

[13] C. Ishikawa, J. A. U. Marasigan, and M. V. C. Caya, "Cloud-based signature validation using CNN inception-resnet architecture," in *2020 IEEE 12th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*. IEEE, 12 2020, pp. 1–6.

[14] J. F. Villaverde, M. D. Ferrer, J. A. T. Macabeo, and J. T. MasilunganManuel, "Classification of cotton fabric, pineapple fabric, and cotton pineapple blend fabric with vgg16 using Keras," in *2023 IEEE 15th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*. IEEE, 11 2023, pp. 1–6.

[15] V. A. M. Luis, M. V. T. Quinones, and A. N. Yumang, "Classification of defects in robusta green coffee beans using yolo," in *4th IEEE International Conference on Artificial Intelligence in Engineering and Technology, IICAIET 2022*. Institute of Electrical and Electronics Engineers Inc., 2022.

[16] M. Li, T. Lv, J. Chen, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei, "Trocr: Transformer-based optical character recognition with pre-trained models," 9 2021.

[17] A. Mortadi, A. Mohamed, A. Talima, A. Alkhattip, A. Ibrahim, A. Osman, and Y. Hifny, "Alnasikh: An Arabic OCR system based on transformers," in *2023 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*. IEEE, 9 2023, pp. 74–81.

Automatic Question Generation from Handwritten Lecture Notes Using TrOCR Text Recognition and T5 Language Processing

Rommel John Ronduen[1], Jan Adrian Manzanero[2], Analyn Yumang[3]
School of Electrical, Electronics, and Computer Engineering
Mapua University
Manila, Philippines 1002
[1]rjhronduen@mymail.mapua.edu.ph
[2]jacmanzanero@mymail.mapua.edu.ph
[3]anyumang@mapua.edu.ph

***Abstract*—This research involves the creation and evaluation of a system that allows for text extraction and automatic question generation (AQG) using a Text-to-Text transformer (T5) and Transformer-based Optical Character Recognition (TrOCR) using keyword extraction from Rapid Automatic Keyword Extraction (RAKE) and the spaCy library. It addresses the lack of OCR implementation with AQG alongside handwritten notes as the basis for question generation. With the use of a Raspberry Pi 5, a web camera, and a touchscreen display, image captures of single-column handwritten notes that only contain textual information generate factoid-type questions. The Stanford Question Answering Dataset (SQuAD) fine-tuned the T5 large language model (LLM) used to facilitate question generation. System evaluation utilized the word error rate (WER) for OCR, while AQG used the ROUGE AND BLEU metrics. The system achieved a WER of 0.40, a ROUGE-1 score of 0.358, and a question validity rate of 68%, resulting in questions with decent context similarity compared to student-made questions. Advanced hardware recommends allowing for larger versions of T5 and TrOCR. Moreover, much emphasis should be placed on OCR processing.**

***Index Terms*— Large Language Model, Optical Character Recognition, Automatic Question Generation, Handwritten Lecture Notes, Raspberry Pi**

I. Introduction

Handwritten lecture notes are often considered the standard method for capturing and facilitating learning. These lecture notes serve as learning artifacts that may contain valuable information, providing the basis for other learning elements. One of these learning elements involves review questions. Quizzes and question banks enable the enforcement of learning across various fields. Since the introduction of artificial intelligence (AI) to education, several advancements have been made that enhance the learning experience of students. Automatic question generation (AQG) is one such advancement. It is achieved using large language models (LLMs) that process various inputs to create questions for student assessment or learning reinforcement.

Despite the feasibility of AQG systems, these cannot process handwritten notes due to the lack of integrated optical character recognition (OCR) [1]. LLMs enable AQG, which has use cases ranging from treebanks [2] to the classification of specific texts [3] [4]. Information-gathering schemes categorize existing AQG frameworks, yet none address handwritten sources [1]. While [5] demonstrated AQG on videos using BERT for named entity recognition (NER) on transcribed audio, textual or handwritten contexts remain unexplored. Other approaches include manual text input with part-of-speech (POS) tagging [6], rule-based methods for programming code [7], and the T5 model fine-tuned on SQuAD for text-based AQG [8]. OCR, defined as image-to-text conversion via convolutional neural networks (CNNs) [9], has been applied to handwritten Baybayin [9], medicinal mushroom classification [10], and expiry date extraction [11]. Through [12], a combination of different neural networks enabled real-time inference, as evaluated via confusion matrices [13] [14] [15]. However, Transformer-based OCR (TrOCR) [16], pre-trained on the IAM Handwriting Database, offers superior paragraph-level text recognition over character-focused CNNs [17]. TrOCR's robustness positions it as a critical enabler for AQG from handwritten lecture notes, addressing the identified gap in media diversity [1].

In bridging the gap for processing handwritten lecture notes for AQG, this research has the general objective of performing AQG from handwritten lecture notes using TrOCR text recognition and T5 language processing. Specifically, this research aims to extract text from handwritten lecture notes using TrOCR and refine such text using the Gemini 1.5 Flash LLM; utilize a fine-tuned base version of T5 on SQuAD for AQG on indexed keywords from spaCy and Rapid Automatic Keyword Extraction (RAKE) on the refined text; utilize a Raspberry Pi 5 within a constructed enclosure with illumination for facilitating the system processes and experimental setup; and lastly, to evaluate the system using the word error rate (WER) for the OCR, and the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) and the Bilingual Language Understanding Evaluation (BLEU) for the comparison of AQG to the student-generated questions.

This system only considers handwritten lecture notes that are single-column, diagram- and equation-free, and written in English on plain letter-size paper. A bare minimum of a single sentence is needed to ensure at least a single reliable question. It is important to note that this system utilizes the base versions of the T5 and TrOCR models, where the former uses SQuAD for the fine-tuning and Gemini 1.5 Flash for the text correction and context completion of OCR-extracted text, as well as the spaCy-Rake keyword extraction for AQG basis. The system requires a single word to be detected to work, producing a minimum of one question up to five. The performance of the Raspberry Pi 5 limits the system's processing capability. A different model for OCR or AQG may lead to differing results, as well as for the text-enhancing model. Lastly, the use of alternative hardware may yield different results.

II. Materials and Methods

A. Hardware development

1) Block Diagram:

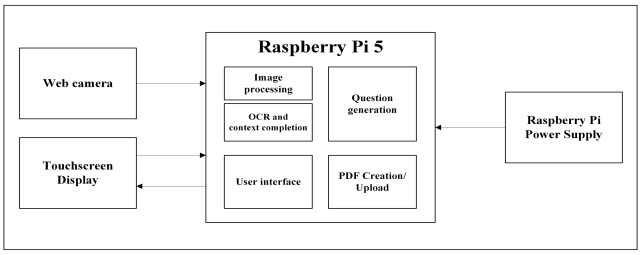


Fig. 1. Hardware block diagram of the system.

Fig. 1 is an illustration of the interconnections of the system. The Raspberry Pi 5 facilitates the system algorithm by utilizing a web camera for capturing handwritten notes and a touchscreen display for user input, thereby dictating the system's operation. The power supply for the Raspberry Pi 5 allows for the delivery of power to the accessory components and main hardware of the system. The system utilized a Logitech C922 web camera, which allowed for autofocus and a 1920 × 1080 pixel capture resolution. Moreover, a Waveshare 800 × 480 touchscreen liquid crystal display (LCD) was used for display purposes while also providing tactile input.

The Docker service deploys the services on the system. Angular serves as the backbone of the front end, which collects images from handwritten notes, video feeds, and interactive buttons. On the other hand, the back end utilized Flask, which enables API handling while also supporting the functions necessary for facilitating the OCR and AQG processes. Lastly, a Page Document Format (PDF) file contained the array of question strings that would be uploaded to Google Drive, where a subsequent download link would be displayed as a quick response (QR) code to the user on the front end.

2) Experimental Setup:

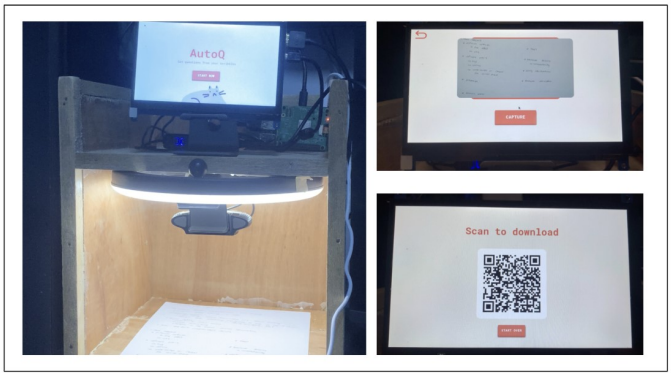


Fig. 2. Constructed prototype and experimental setup.

The constructed prototype is shown in Fig. 2. An experimental setup was conducted where the system requires handwritten notes to be entered into the interior of the prototype. An internet connection is needed for the experimental setup. The user is then guided by the front end to capture the handwritten notes. Two halves are captured for the notes, where the first half is the top half and the second half is the bottom half. The user is then prompted to confirm the captured notes for the system to proceed with the OCR operation. The system then proceeds with the OCR and AQG operations. The user is then prompted to confirm the generated questions, allowing the system to proceed with developing the PDF. The user is then prompted to download the PDF file for the generated questions.

B. Software development

1) Algorithm Pipeline:

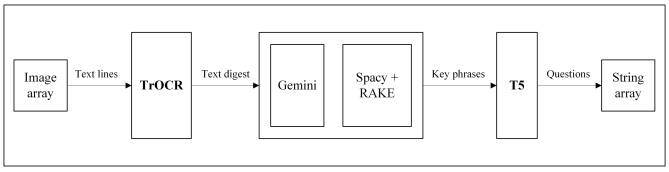


Fig. 3. Algorithm pipeline.

Shown in Fig. 3 is the illustration of the step-by-step flow of information on the system algorithm. An image array in the system's memory consists of image captures of the handwritten lecture notes, represented as OpenCV image objects. These images undergo a series of fast denoising, thresholding, and dilation methods to detect text lines. These text lines are stored in another array where TrOCR performs batch inferences to attempt extraction from the sources. spaCy and RAKE collected relevant keywords and phrases as a basis for question generation, which was then fed to the T5 model for further processing. These questions are then stored in an array for PDF creation.

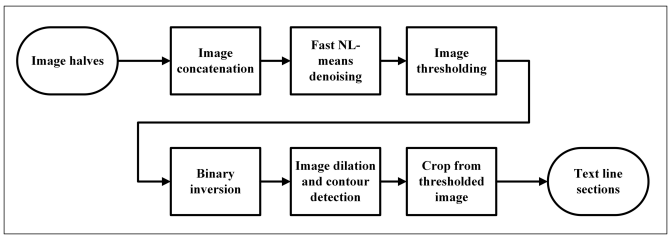


Fig. 4. Text line detection and extraction process.

Revealed in Fig. 4 is the step-by-step procedure for the process of retrieving text lines from a given document. It starts with the receipt of two half-scans of a given document. Image concatenation involves combining the two images to recreate the full document. Fast non-local (NL) denoising and image thresholding are used to convert the document scan into a more readable document while preserving the shape of the handwritten text. Image binarization and horizontal dilation enable the revelation of text lines throughout the document. Finally, the detected text lines from the contours are cropped from the original image.

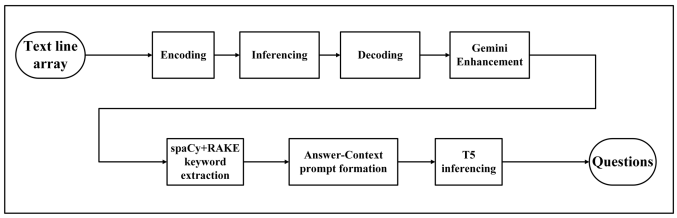


Fig. 5. TrOCR and T5 processing.

Fig. 5 presents the detailed operation of TrOCR in the system. Coming from the extracted text line images from the process in Fig. 4, batch encoding, inference, and decoding were done on the array of text lines using the encoder-decoder framework provided by TrOCR. After the extraction procedure, it is prompted to Gemini, where error correction is attempted from the OCR extraction. Coming from an enhanced text in Fig. 5, the spaCy library and the RAKE algorithm were used to find the relevant keywords and phrases from the text as the basis for the questions. Consequently, answer-context pairs are used as prompts that are inputted to the T5 model that generates the questions in an array.

2) Model Training:

The T5 base model was fine-tuned on the SQuADv1.1 dataset for the AQG operation, using a learning rate of 3 × 10^-5 for eight epochs with the Sequence2Sequence trainer. The SQuADv1.1 dataset was loaded using the datasets library, where the chosen input features were the context and the question. The output feature was the answer for the given pair of inputs.

C. Data Collection

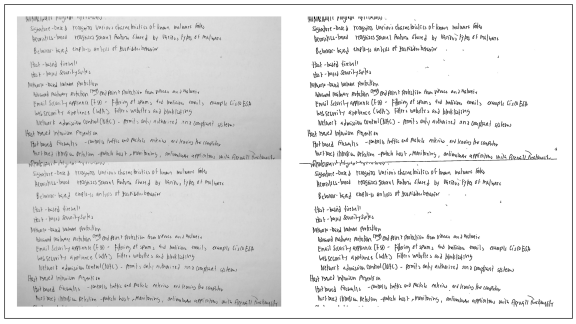


Fig. 6. Sample from the test set. Combined image vs. preprocessed image

Shown in Fig. 6 is a sample from the test set containing 70 handwritten notes collected from undergraduate classes on computer networks and cybersecurity. These notes were written in English, single-column, and diagram-free. A total of 1800 individual text lines were extracted from the notes for the evaluation of the OCR operation through manual transcription and were saved in a comma-separated value (CSV) format for analysis. As part of the system operation on the 70 handwritten notes, 350 questions (five questions per page) were produced for evaluation.

Moreover, 135 questions were collected from students who took notes during the class. The top five most common keywords from the overall extracted text (MAC, Ethernet, address, data, and frame) were utilized as the basis for the questions, alongside a summary written from the extracted notes that contained such keywords.

D. Testing and Evaluation

TABLE I
TESTING AND EVALUATION TABLE FOR OCR.

	Student *(sample)*	Model *(sample)*	WER *(sample)*
1	Establishing a robust operating system security baseline is crucial	Establishing a robust operating system security baseline is crucial	0.32
...	...	...	...
70	Malware detection employs various techniques, including signature-based methods that recognize characteristics of	Malware detection employs multiple techniques, including signature-based, which recognizes various characteristics of	0.49
		Average	**0.39**

Shown in Table I is the evaluation of the OCR operation using the metric of word error rate (WER) on the 70 handwritten notes. Testing is done through the collection of the transcribed text versus the inference of the model. The WER is calculated using the formula:

$$WER = \frac{S + D + I}{N} \quad (1)$$

Where S is equivalent to several substitutions, D is the number of absent words, I *is* for new words, and N is the total number of words in the reference text [9].

TABLE II
TESTING AND EVALUATION TABLE FOR QUESTION VALIDITY

	Question *(sample)*	Validity *(sample)*
1	What work is the Ethernet?	Valid
...	...	...
350	What does MAC do?	Invalid
	Total	**299**

Shown in Table II is the testing and evaluation table for the validity of the systems. The questions were generated from the 70 handwritten notes and were evaluated for grammar, syntax, and correlation to the answer by the professor of the cybersecurity course. The question validity rate is computed as follows:

$$\text{Question Validity Rate} = \frac{\text{Valid Questions}}{\text{Total Questions}} \times 100\% \quad (2)$$

TABLE III
TESTING AND EVALUATION TABLE FOR AQG.

	Student *(sample)*	Model *(sample)*	ROUGE *(sample)*	BLEU *(sample)*
1	What works alongside MAC address to ensure seamless data transfer.	What does MAC work alongside to ensure seamless data transfer?	0.312	0.433
...	...	...	...	...
135	What is the traditional technology for connecting devices (LAN) or vast area networks?	What other network uses MAC? Connected devices?	0.662	0.255
		Average	**0.557**	**0.192**

Shown in Table III is the evaluation of the AQG operation using the ROUGE and the BLEU scores. Data collection allowed for the accumulation of 135 questions from students. Using the top five keywords from the handwritten lecture notes, they are compared with the model-generated results based on the basis of the keyword and context. ROUGE and BLEU are computed as follows:

$$ROUGE = \frac{1}{N}\sum_{i=1}^{N}\frac{2 \times (P \times R)}{(P + R)} \quad (3)$$

$$BLEU = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{r}\sum_{i=1}^{r}\min\left(\sum_{i=1}^{r}\text{count}_{\text{clip}}(ngram)\right) \quad (4)$$

Where P is the precision, R is the recall, N is the number of notes, r is the number of reference questions, and $\text{count}_{\text{clip}}$ is the count of the clipped n-grams, summed over the target sentences with the index$i$.

III. RESULTS AND DISCUSSION

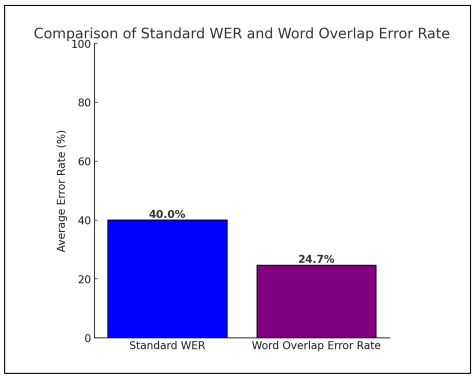


Fig. 7. Computed WER and WOER

In Fig. 7, it was shown that a WER of 0.40 or 40% was realized from the implementation of the TrOCR model with a word overlap error rate (WOER) of 0.247 or 24.7% that measures the completeness of the inference versus the reference. The relatively moderate magnitude of the WER was realized due to the natural misalignment of the text lines from the handwritten notes. Moreover, the lack of use-case-specific fine-tuning for TrOCR may have contributed to the magnitude of the word error rate (WER). Nevertheless, the 60% accuracy allowed enough room for Gemini 1.5 Flash to correct and bridge the gaps in the text digest to form valid questions.

In terms of the model-generated questions, it was calculated that, on average, there were 9.87 or approximately 10 words in every generated question.

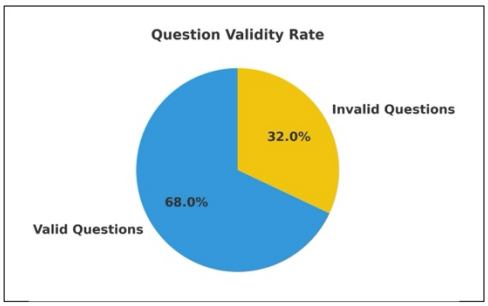


Fig. 8. Question Validity Rate Pie Chart

Fig. 8 revealed that **the system-generated questions were valid 68% of the time. Out of 350 questions, 238 were deemed valid in terms of coherence, grammar, and relevance to the subject matter.** The remaining invalidity of 32% was attributed to possibly two aspects. First, the T5 model may require additional parameters, such as those of the large or extra-large (XL) model, to generate more effective questions. The second is that some keywords may not have been corrected by the Gemini 1.5 Flash model, which may have led to the propagation of errors in the generated questions.

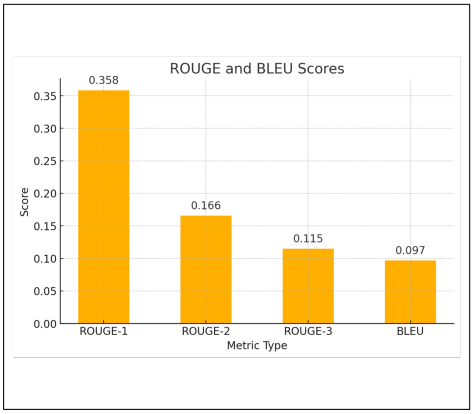


Fig. 9. ROUGE scores and BLEU evaluation

Shown in Fig. 9 is the evaluation of the AQG operation using the ROUGE scores and the BLEU scores. It is essential to note that the questions posed by students often resemble verbatim copies of the basic context. **A ROUGE-1 score of 0.358** meant that the system decently captures the relevant keywords in the same way as students do. Moreover, the ROUGE scores of 0.166 and 0.115 indicated that the model-based questions differed from the students' syntax, as many students verbatim copied the structure of the context, compared to the dynamic outputs of the system. Since it was calculated that there are 9.87 or 10 average words per generated question, **each question may have up to 4 out of 10 words matched from student-made questions**. Lastly, the BLEU score of 0.097 indicated that the system could grasp context but not the logical structure of the students due to high lexical variability. Ultimately, this means that the questions generated by the system deviate from a fixed structure but do not impact validity (Fig. 8).

IV. CONCLUSION AND RECOMMENDATIONS

It was realized that the AQG from handwritten notes through TrOCR was possible with a 68% question validity and 40% word error rate. The questions generated approximately have four matching keywords from student-made questions, as shown by the computed ROUGE-1 to ROUGE-3 scores of 0.351, 0.166, and 0.115, and a BLEU score of 0.097, indicating little similarity since the values are closer to 0.

It is recommended that further preprocessing of the image captures be performed to improve the text extraction. Additionally, the use of larger versions of TrOCR and T5 is recommended, along with the utilization of more powerful hardware to enhance system processing capabilities and an improved camera for higher image resolutions. It is also recommended that alternative evaluation methods be utilized for this AQG implementation, such as similarity or semantic analyses.

REFERENCES

[1] A. Arbaaeen and A. Shah, "Natural language processing based question answering techniques: A survey," in *7th IEEE International Conference on Engineering Technologies and Applied Sciences, ICETAS 2020*. Institute of Electrical and Electronics Engineers Inc., 12 2020.

[2] D. A. Padilla, A. J. M. Mesina, and E. J. A. Perez, "English treebank of Mapua University smart interactive voice response system," in' *2020 IEEE 12th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*. IEEE, 12 2020, pp. 1–6.

[3] D. A. Padilla, N. K. U. Vitug, and J. B. S. Marquez, "Deep learning approach in gregg shorthand word to english-word conversion," in *2020 IEEE 5th International Conference on Image, Vision and Computing (ICIVC)*. IEEE, 7 2020, pp. 204–210.

[4] A. N. Yumang, M. J. F. D. Cru, and M. C. Q. Jasmin, "Relationship strength of handwriting consistency and number of local features with the accuracy of a handwritten forgery detection model," in *2022 IEEE 14th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management, HNICEM 2022*. Institute of Electrical and Electronics Engineers Inc., 2022.

[5] Y. Y. Ou, S. W. Chuang, W. C. Wang, and J. F. Wang, "Automatic multimedia-based question-answer pairs generation in the computer-assisted healthy education system," in *2022 10th International Conference on Orange Technology, ICOT 2022*. Institute of Electrical and Electronics Engineers Inc., 2022.

[6] M. Moron, J. Scocozza, L. Chiruzzo, and A. Rosa, "A tool for automatic question generation for teaching English to beginner students," in *Proceedings - International Conference of the Chilean Computer Science Society, SCCC*, vol. 2021-November. IEEE Computer Society, 2021.

[7] H. Gaur, D. K. Tayal, and A. Jain, "Viqg: Web tool for automatic question generation from code for viva preparation," in *Proceedings of 2023 26th Conference of the Oriental COCOSDA International Committee for the Co-ordination and Standardization of Speech Databases and Assessment Techniques, O-COCOSDA 2023*. Institute of Electrical and Electronics Engineers Inc., 2023.

[8] D. C. Tsai, A. Y. Huang, O. H. Lu, and S. J. Yang, "Automatic question generation for repeated testing to improve student learning outcome," in *Proceedings - IEEE 21st International Conference on Advanced Learning Technologies, ICALT 2021*. Institute of Electrical and Electronics Engineers Inc., 7 2021, pp. 339–341.

[9] A. M. P. Ligsay, J. B. Rivera, and J. F. Villaverde, "Optical character recognition of baybayin writing system using yolov3 algorithm," in *2022*

IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET). IEEE, 9 2022, pp. 1–5.

[10] M. J. Y. Sutayco and M. V. C. Caya, "A comparative study of mobilenetv2 and vgg-16 convolutional neural network architectures for identification of medicinal mushrooms." Institute of Electrical and Electronics Engineers (IEEE), 7 2024, pp. 1–6.

[11] C. O. Manlises, D. A. Padilla, J. B. Santos, and P. A. Adviento, "Expiration identification of canned goods using convolutional neural network," in *2024 7th International Conference on Information and Computer Technologies (ICICT)*. IEEE, 3 2024, pp. 173–177.

[12] A. D. R. Calimag, D. A. Padilla, and C. O. Manlises, "Checkout system with object detection using nvidia jetson nano and Raspberry Pi," in *2023 IEEE 5th Eurasia Conference on IOT, Communication and Engineering (ECICE)*. IEEE, 10 2023, pp. 168–171.

[13] C. Ishikawa, J. A. U. Marasigan, and M. V. C. Caya, "Cloud-based signature validation using CNN inception-resnet architecture," in *2020 IEEE 12th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*. IEEE, 12 2020, pp. 1–6.

[14] J. F. Villaverde, M. D. Ferrer, J. A. T. Macabeo, and J. T. MasilunganManuel, "Classification of cotton fabric, pineapple fabric, and cotton pineapple blend fabric with vgg16 using Keras," in *2023 IEEE 15th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*. IEEE, 11 2023, pp. 1–6.

[15] V. A. M. Luis, M. V. T. Quinones, and A. N. Yumang, "Classification of defects in robusta green coffee beans using yolo," in *4th IEEE International Conference on Artificial Intelligence in Engineering and Technology, IICAIET 2022*. Institute of Electrical and Electronics Engineers Inc., 2022.

[16] M. Li, T. Lv, J. Chen, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei, "Trocr: Transformer-based optical character recognition with pre-trained models," 9 2021.

[17] A. Mortadi, A. Mohamed, A. Talima, A. Alkhattip, A. Ibrahim, A. Osman, and Y. Hifny, "Alnasikh: An Arabic ocr system based on transformers," in *2023 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*. IEEE, 9 2023, pp. 74–81.

Automatic Question Generation from Handwritten Lecture Notes on KeyBERT-indexed T5-TrOCR Pipeline Using BERTscore Evaluation

Rommel John Ronduen[1], Jan Adrian Manzanero[2], Analyn Yumang[3]
School of Electrical, Electronics, and Computer Engineering
Mapua University
Manila, Philippines 1002
[1]rjhronduen@mymail.mapua.edu.ph
[2]jacmanzanero@mymail.mapua.edu.ph
[3]anyumang@mapua.edu.ph

***Abstract*—This research improves on an automatic question generation (AQG) system that utilizes optical character recognition (OCR) from handwritten lecture notes as a context source for generating questions. Such a system uses the Text-to-Text transformer (T5) for AQG and Transformer-based Optical Character Recognition (TrOCR) for OCR, leveraging Gemini version 1.5 large language model (LLM) for text correction and enhancement. A version of the Bidirectional Encoder Representations from Transformers (BERT), known as KeyBERT, was utilized to extract relevant phrases and keywords from the information, providing a basis for question generation. The system utilizes a Raspberry Pi 5 equipped with a Logitech C922 web camera and a touchscreen liquid crystal display (LCD). A web application was developed using Angular and Flask for prototyping, allowing for the AQG process to facilitate experimentation. The system achieved 85.1% question validity, with 64.1% precision, 59.77% recall, and an F1 score of 61.67, utilizing BERTscore semantic analysis on undergraduate-level computer networks and cybersecurity questions compared to student-made questions. This system generates questions with decent context capture but with recommended improvements in hardware, model parameter size, and image preprocessing techniques. This system generates questions with decent context capture, but with recommended improvements in hardware, model parameter size, and image preprocessing techniques.**

***Index Terms*—Automatic Question Generation, Optical Character Recognition, BERT, T5, Raspberry Pi**

I. Introduction

Handwritten lecture notes are still widely used in educational institutions. Since the advent of artificial intelligence (AI), education has become the primary focus of AI research. AI has been used to automate several processes, one of which is the generation of learning materials, such as questions in the form of quizzes. Literature defines automatic question generation (AQG) as the process of generating questions from a given text.

No AQG implementation involved the process of optical character recognition (OCR) for extracting text content from handwritten lecture notes with semantic evaluation. For instance, [1] provided AQG from programming source code where a user interface allows for user text input and shows the generated questions. In a correlational study of AQG to student learning [2], an AQG implementation used existing text files with the Bidirectional Encoder Representations from Transformers (BERT) for keyword extraction and the T5 model for the generation of questions. Another AQG implementation utilized video media [3], where audio recognition led to transcript generation, which was then used for AQG through the Bidirectional and Auto-Regressive Transformer (BART). To address the lack of OCR in existing AQG implementations, various types of models can be employed, which are based on convolutional neural networks (CNNs). CNNs can be used for general classification [4] [5]. For instance, MobileNetv2 and VGG16 are types of CNNs that can classify medicinal mushrooms [6]. These CNNs comprise different layers that perform transformations on the input image, resulting in the extraction of features used for inference [7] [8]. In evaluating CNNs, confusion matrices provide an illustration and computation of accuracy [9] [10]. OCR can be made possible through CNNs using proper image capture. For instance, Baybayin characters were recognized correctly [11]. However, OCR technology has advanced to the point where transformers are now used through the Transformer-based Optical Character Recognition (TrOCR) model. This model utilizes an encoder-decoder framework to parse images of text lines and produce corresponding strings [12]. Due to its pre-trained availability and seamless implementation in Python, this model was chosen for this research. During the process of question generation, the researchers discovered that BERT and T5. With BERT, the model comes pre-trained and can be utilized in text classification and indexing [13] [14]. BERT comes pre-trained, meaning it is immediately capable of extracting semantic features from text for inferencing [15]. Due to the fine-tuning offered by BERT, many versions of BERT have been realized that serve different specialized purposes [16]. KeyBERT is a variant of BERT that was fine-tuned for extracting relevant words and phrases in a paragraph through the cosine similarity of inferred candidate structures in the text. KeyBERT has been utilized in the implementation of AQG [17] to produce multiple-choice questions, where KeyBERT indexes the choices. Apart from generative purposes, BERT can be used for

evaluation through BERTscore semantic analysis. Since BERT can be used to understand the meaning between a reference and generated text result, the score can infer the similarity among sentences and provide recall and precision metrics based on the meaning of the sentences [18], which *n*-gram overlap methods may not realize. In total, this research addresses the lack of OCR in AQG implementations by utilizing TrOCR and Gemini 1.5 Flash for text enhancement and extraction from handwritten lecture notes while employing T5 and KeyBERT for AQG with semantic evaluation through BERTscore. TrOCR and Gemini 1.5 Flash for text enhancement and extraction from handwritten lecture notes, while employing T5 and KeyBERT for AQG with semantic evaluation through BERTscore.

In general, this research aims to enable AQG from handwritten lecture notes using a KeyBERT-indexed T5-TrOCR algorithm pipeline, evaluated through BERTscore semantic analysis. More specifically, this research seeks to utilize KeyBERT indexing for extracting relevant keywords and phrases coming from the extracted text through TrOCR and Gemini 1.5 Flash as the basis for AQG through the base model T5 fine-tuned on the Stanford Question Answering Dataset (SQuADv1.1). Moreover, this research aims to utilize a Raspberry Pi 5 with an 8-gigabyte memory version, alongside a Logitech C922 web camera and a Waveshare touchscreen LCD, to facilitate the necessary inputs and outputs for the system through a constructed enclosure with proper lighting. Lastly, this research aims to evaluate the quality of AQG through KeyBERT semantic analysis and assess the validity of questions via manual analysis.

This research is limited by the consideration of only single-column, diagram-free, and equation-free handwritten lecture notes in English, with no erasures, on strictly letter-sized plain sheet paper. Moreover, the system is limited to the use of the T5TrOCR pipeline, utilizing the base versions, and also utilizing the SQuADv1.1 dataset for fine-tuning. It is also important to note that the system is limited to facilitating the processes in the hardware of the 8-gigabyte version of the Raspberry Pi 5. The use of a higher parameter count for the T5 and TrOCR models, along with the use of a different dataset for fine-tuning, may yield differing results. Lastly, the utilization of a different hardware setup may also lead to alternative, if not better, results in terms of processing time. results in terms of processing time.

II. Materials and Methods

A. Hardware Development

1) Hardware System Block Diagram:

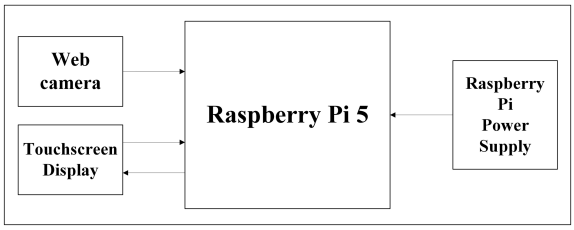


Fig. 1. Hardware system.

The system block diagram is shown in Fig. 1. The system is composed of a Raspberry Pi 5, a USB web camera, and a touchscreen monitor. The processing unit of the system is the Raspberry Pi 5. The web camera is used for capturing images of handwritten lecture notes. The touchscreen monitor is used for displaying the generated questions. The system is enclosed in a box with proper illumination to facilitate the processes.

2) Experimental Setup:

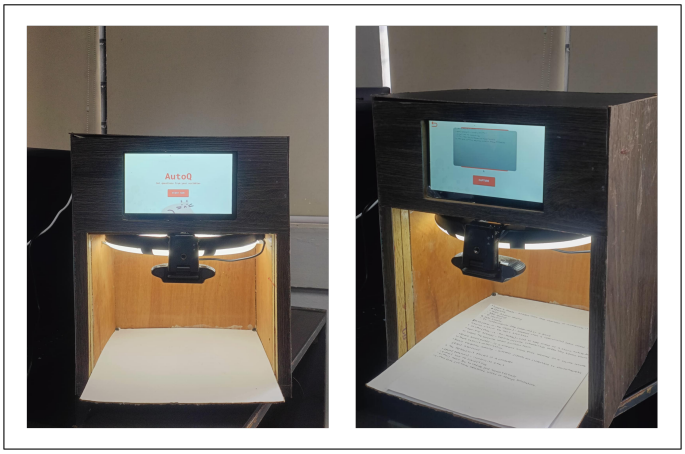


Fig. 2. Actual experimentation setup.

Fig. 2 shows the actual setup in the experimentation. The system is enclosed within a constructed wooden box with adequate illumination. The Raspberry Pi 5 is connected to the web camera and the touchscreen monitor. The web camera is used for capturing images of handwritten lecture notes. The touchscreen monitor is used to guide the user through the system's processes.

B. Software Development

1) System Flowchart:

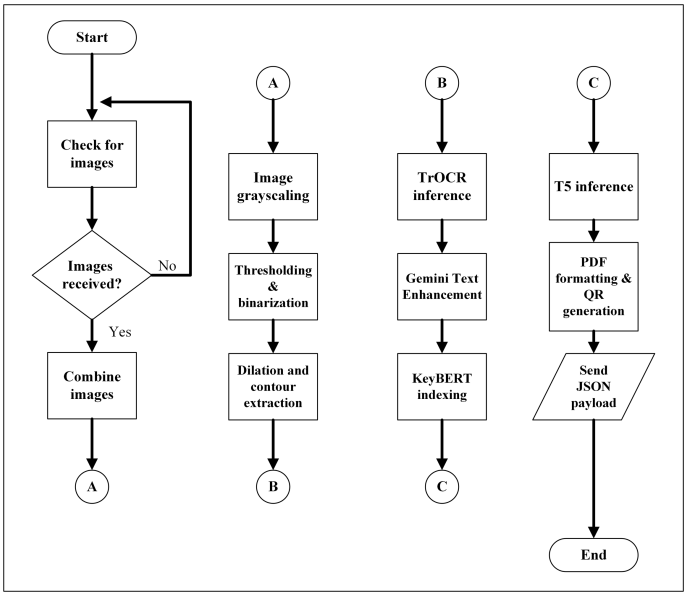


Fig. 3. Algorithm process.

Fig. 3 is the diagram for the system algorithm. The system begins by capturing images of the top and bottom halves of the notes. The photo captures are then combined and undergo a series of image transformations that extract the text lines. These text lines were input to the TrOCR for inference, and the extracted text was then input to the T5 for AQG. The generated questions are formatted in Portable Document Format (PDF) and uploaded to Google Drive, where a quick response (QR) code is generated for the user to download the PDF to their device. A JavaScript Object Notation (JSON) payload is then sent to the front end of the system for the notification to be displayed on the touchscreen.

2) T5 AQG:

In this research, T5 was fine-tuned to the SQuADv1.1 dataset. The input features considered were the context and the chosen answer phrase or keyword. The output feature was the question generated from the context and the chosen answer phrase/keyword.

C. Data Gathering

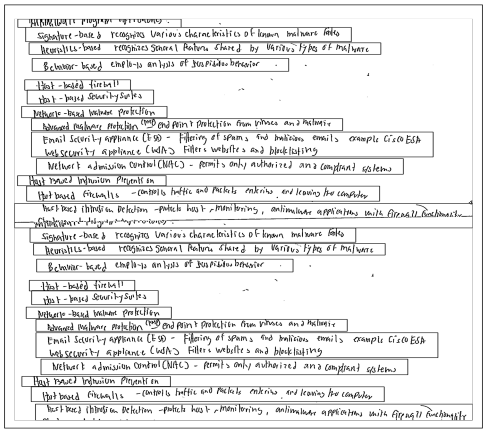


Fig. 4. Sample processed note capture.

A sample capture from the procured testing dataset is shown in Fig. 4. A total of 70 handwritten lecture notes were collected from an undergraduate class on computer networking and cybersecurity. These notes were input into the system, which will result in the generation of 350 questions for validity testing.

Apart from the notes, the researchers collected questions from students based on a summary discussion that identified the top five most common keywords (MAC, address, Ethernet, data, frame) in the students' notes. In a class of 27 students, each was asked to formulate questions that answered the five common keywords. A total of 135 questions were collected from the students.

D. Testing and Evaluation

TABLE I
TESTING AND EVALUATION TABLE FOR QUESTION VALIDITY.

	Question *(sample)*	Validity*(sample)*
1	What does MAC stand for?	Valid
...	...	...
350	What is network?	Invalid
	Total	**271**

Revealed in Table, I is the testing and evaluation of the 350 questions generated by the system as a byproduct of its intake of the 70 handwritten notes from the data collection. Here, the professor of the cybersecurity course evaluated the questions for grammar, syntax, and correlation to answer. After calculating the total number of valid questions, it is divided by the total number of generated questions to obtain the question validity rate. Mathematically:

$$\text{Validity} = \frac{\text{Valid questions}}{\text{Total questions}} \times 100\% \qquad (1)$$

TABLE II
TESTING, AND EVALUATION TABLE FOR AQG.

	Student *(sample)*	Model *(sample)*	Recall *(sample)*	Precision *(sample)*	F1 *(sample)*
1	This is used to connect devices on a local area network via wired Connections.	MAC works alongside what to ensure seamless data transfer between connected devices.	0.55	0.62	0.45
...	...	...	...	...	...
135	What must be a certain characteristic of the MAC address to allow proper identification	What is assigned to each device?	0.32	0.77	0.52
		Average	**0.15**	**0.33**	**0.66**

Shown in Table II is the testing and evaluation table for AQG evaluation, where recall, precision, and F1 scores are calculated

for each of the 135 questions from the students using BERTscore.

$$\text{Recall} = \frac{1}{|x|}\sum_{\tilde{x_j}\in\tilde{x}} \max \mathbf{x}_i^T \tilde{\mathbf{x}}_j \tag{2}$$

$$\text{Precision} = \frac{1}{|\tilde{x}|}\sum_{\tilde{x_i}\in\tilde{x}} \max \mathbf{x}_i^T \tilde{\mathbf{x}}_j \tag{3}$$

$$\text{F1} = 2\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}} \tag{4}$$

Where x is the reference sentence, $\tilde{x}$ is the tokenized inference sentence, and $\boldsymbol{x}_i^T \tilde{\boldsymbol{x}}_j$ is the cosine similarity of the two sentences with i and j being the indices summed over the sentence embeddings. The recall quantity takes the average of the best similarities across the reference sentences. As for the precision, it also$j$exact amounts, but based on the inference sentences $\tilde{x}$. Lastly, the F1 score computes the harmonic mean across (2) and (3) as a form of equalized score between precision and recall, but based on the inference sentences $\tilde{x}$. Lastly, the F1 score computes the harmonic mean across (2) and (3) as a form of equalized score between precision and recall.

III. Results and Discussion

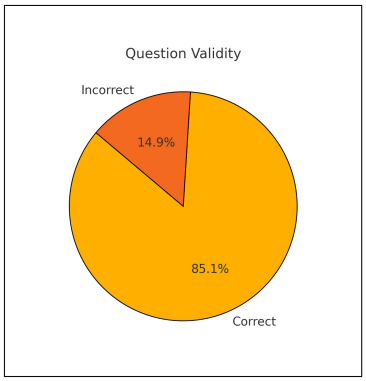


Fig. 5. Pie chart for valid questions versus invalid.

As shown in Fig. 5, 85.1% of the questions generated out of the test set were valid in terms of syntax and its correlation to the answer. It means that 298 out of the 350 questions were able to lead to the designated answer while providing proper structure and grammar. The use of KeyBERT likely enabled the T5 model to generate questions, such as how SQuAD formed its questions. The remaining 14.9% or 52 questions were considered invalid due to grammar, invalid structure, and misalignment with the answer it suggests. This would have been due to the lack of parameters in T5's formulation of the questions, as the researchers have noticed in the inferences that errors in invalid questions involve the absence of linking verbs or the revelation of the answer within the question.

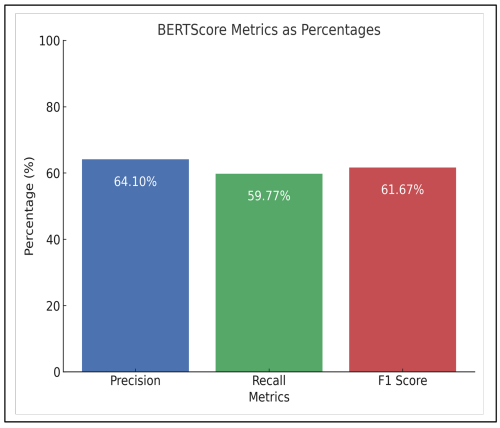


Fig. 6. BERTscore metrics result.

Fig. 6 presents the metrics produced by BERTscore. The system was 64.10% precise, 59.77% recall, and had an F1 score of 61.67%. Generated questions decently contain the original context from the reference. During runtime, three out of the five generations would have decent adherence to the basic text. During analysis of the generated questions, some questions lack the relevant keywords from the reference question, as well as the usual errors discussed in question validity. Moreover, some students utilized different WH questions compared to the inference. Lastly, it was noted that the questions generated by the system were considerably shorter, ranging from 9 to 14 words, whereas the reference questions could be up to 17 words in length.from 9 to 14 words, whereas the reference questions could be up to 17 words.

IV. Conclusion and Recommendations

It was concluded that the system utilizing the T5 and TrOCR pipeline was capable of performing AQG from handwritten lecture notes with satisfactory validity and relevance, but was unpredictable in terms of lexical structure.

It is recommended that further research on this implementation use better hardware. This will lead to the use of higher-parameter-count versions of T5 and TrOCR, which will improve the system's operation. Lastly, it is recommended that considerable attention be given to the preprocessing of image captures of handwritten notes to enhance word capture accuracy through text enhancer LLMs or by utilizing an alternative framework for OCR. word capture accuracy through text enhancer LLMs or by utilizing an alternative framework for OCR.

References

[1] H. Gaur, D. K. Tayal, and A. Jain, "Viqg: Web tool for automatic question generation from code for viva preparation," in *Proceedings of 2023 26th Conference of the Oriental COCOSDA International Committee for the Co-ordination and Standardization of Speech Databases and Assessment Techniques, O-COCOSDA 2023*. Institute of Electrical and Electronics Engineers Inc., 2023.

[2] D. C. Tsai, A. Y. Huang, O. H. Lu, and S. J. Yang, "Automatic question generation for repeated testing to improve student learning outcome," in *Proceedings - IEEE 21st International Conference on Advanced*

Learning Technologies, ICALT 2021. Institute of Electrical and Electronics Engineers Inc., 7 2021, pp. 339–341.

[3] Y. Y. Ou, S. W. Chuang, W. C. Wang, and J. F. Wang, "Automatic multimedia-based question-answer pairs generation in the computer-assisted healthy education system," in *2022 10th International Conference on Orange Technology, ICOT 2022*. Institute of Electrical and Electronics Engineers Inc., 2022.

[4] C. O. Manlises, D. A. Padilla, J. B. Santos, and P. A. Adviento, "Expiration identification of canned goods using a convolutional neural network," in *2024 7th International Conference on Information and Computer Technologies (ICICT)*. IEEE, 3 2024, pp. 173–177.

[5] A. D. R. Calimag, D. A. Padilla, and C. O. Manlises, "Checkout system with object detection using Nvidia Jetson Nano and Raspberry Pi," in *2023 IEEE 5th Eurasia Conference on IOT, Communication and Engineering (ECICE)*. IEEE, 10 2023, pp. 168–171.

[6] M. J. Y. Sutayco and M. V. C. Caya, "A comparative study of mobilenetv2 and vgg-16 convolutional neural network architectures for identification of medicinal mushrooms," in *2023 IEEE 15th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management, HNICEM 2023*. Institute of Electrical and Electronics Engineers Inc., 2023.

[7] A. P. S. Estrella, F. M. O. Lucban, and M. V. C. Caya, "Image recognition of different hamster breeds using convolutional neural networks," in *6th IEEE International Conference on Artificial Intelligence in Engineering and Technology, IICAIET 2024*. Institute of Electrical and Electronics Engineers Inc., 2024, pp. 597–602.

[8] M. G. Wata and J. F. Villaverde, "Bubble-sheet assessment checker with test grader using computer vision through Raspberry Pi," in *2023 IEEE 6th International Conference on Computer and Communication Engineering Technology, CCET 2023*. Institute of Electrical and Electronics Engineers Inc., 2023, pp. 137–142.

[9] G. L. B. Bacalla, E. N. C. Corate, and A. N. Yumang, "Jackfruit maturity classification using uniform local binary pattern and support vector machine," in *2023 IEEE 5th Eurasia Conference on IOT, Communication and Engineering, ECICE 2023*. Institute of Electrical and Electronics Engineers Inc., 2023, pp. 605–610.

[10] J. M. Baguisi, B. R. S. Buenaventura, and A. N. Yumang, "The effect of data augmentation and padding of the image dataset on detection of black sigatoka disease on banana leaves using shufflenet v2 cnn architecture," in *Proceedings - 2024 7th International Conference on Information and Computer Technologies, ICICT 2024*. Institute of Electrical and Electronics Engineers Inc., 2024, pp. 277–282.

[11] A. M. P. Ligsay, J. B. Rivera, and J. F. Villaverde, "Optical character recognition of baybayin writing system using YOLOv3 algorithm," in *4th IEEE International Conference on Artificial Intelligence in Engineering and Technology, IICAIET 2022*. Institute of Electrical and Electronics Engineers Inc., 2022.

[12] M. Li, T. Lv, J. Chen, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei, "Trocr: Transformer-based optical character recognition with pre-trained models," 9 2021.

[13] V. I. D. Rosario, B. D. P. Fernandez, and D. A. Padilla, "Email spam classification using distillery," in *2023 IEEE 15th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*. IEEE, 11 2023, pp. 1–6.

[14] J. Mingua, D. Padilla, and E. J. Celino, "Classification of fire-related tweets on Twitter using bidirectional encoder representations from transformers (Bert)," in *2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*. IEEE, 11 2021, pp. 1–6.

[15] Y. Liu, H. Huang, J. Gao, and S. Gai, "A study of Chinese text classification based on a new type of Bert pre-training," in *Proceedings - 2023 5th International Conference on Natural Language Processing, ICNLP 2023*. Institute of Electrical and Electronics Engineers Inc., 2023, pp. 303–307.

[16] C. Patra, D. Giri, T. Maitra, and B. Kundu, "A comparative study on detecting phishing URLs leveraging pre-trained BERT variants," in *2024 6th International Conference on Computational Intelligence and Networks (CINE)*. IEEE, 12 2024, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/10881521/

[17] P. K. Talupuri, M. G. Kumar, K. Lavanya, G. Chetan, and N. P. Challa, "Smart MCQ generation using keybert and Bert-based models," in *2024 3rd Edition of IEEE Delhi Section Flagship Conference (DELCON)*. IEEE, 11 2024, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/10866615/

[18] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with Bert," 4 2019. [Online]. Available: http://arxiv.org/abs/1904.09675

APPENDIX A: FIGURES

APPENDIX A: FIGURES

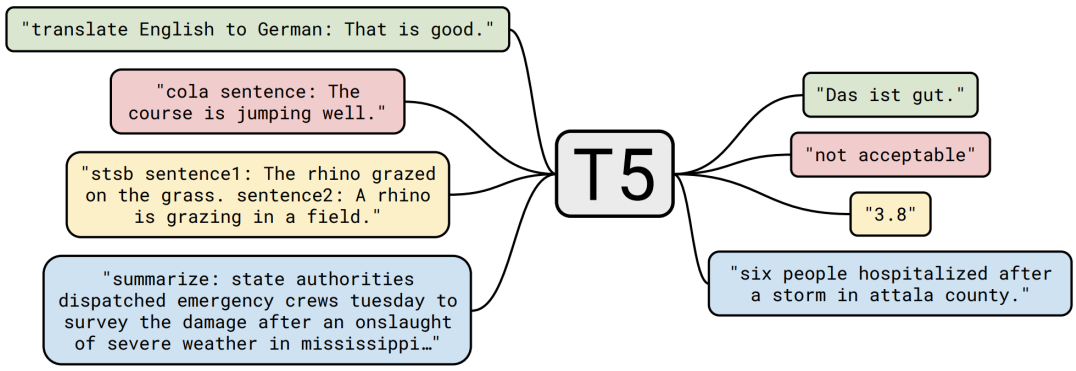


Figure 2.1: T5 model framework (sourced from [9])

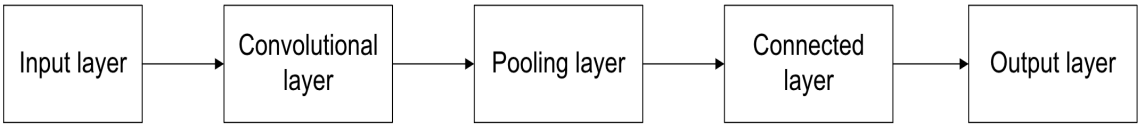


Figure 2.2: A basic representation of a CNN model

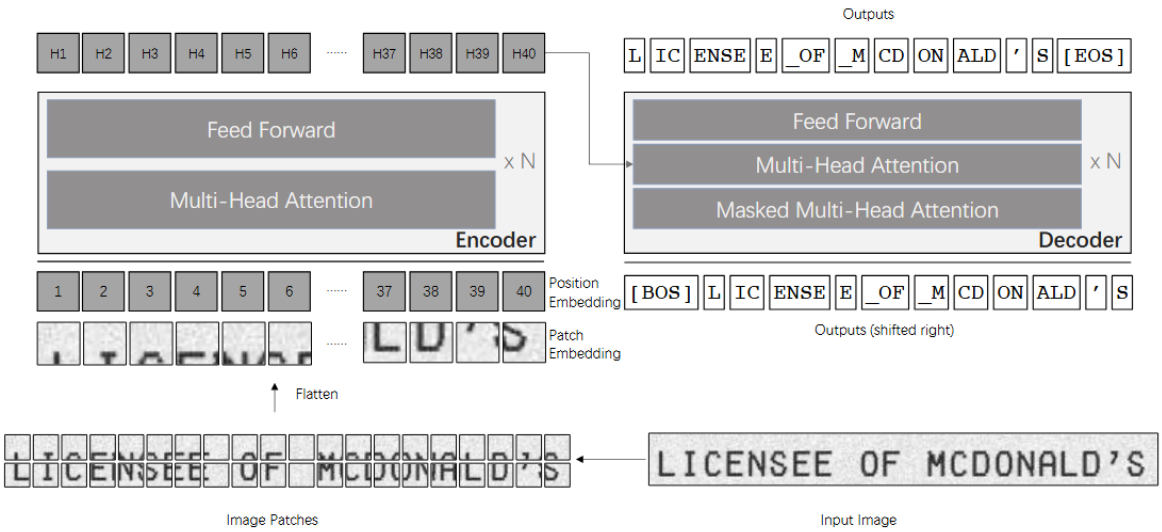


Figure 2.3: Model architecture of TrOCR (sourced from [23])

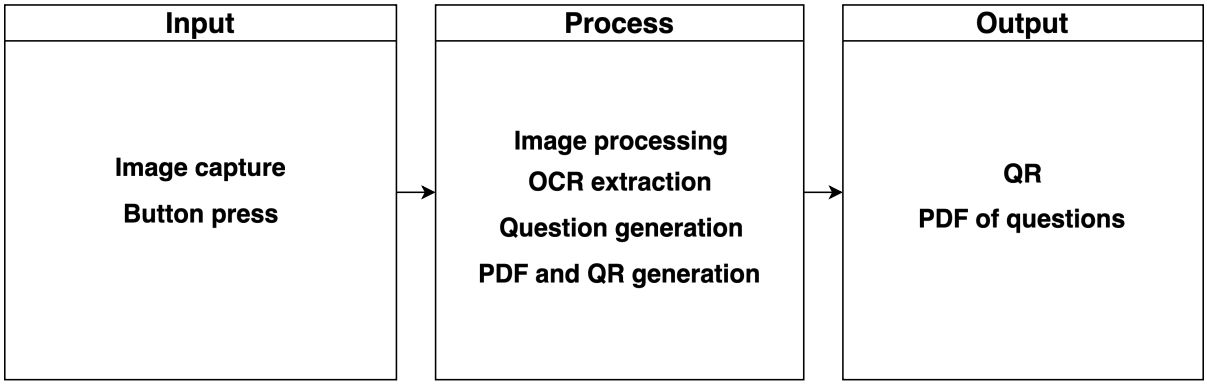


Figure 3.1: Conceptual framework

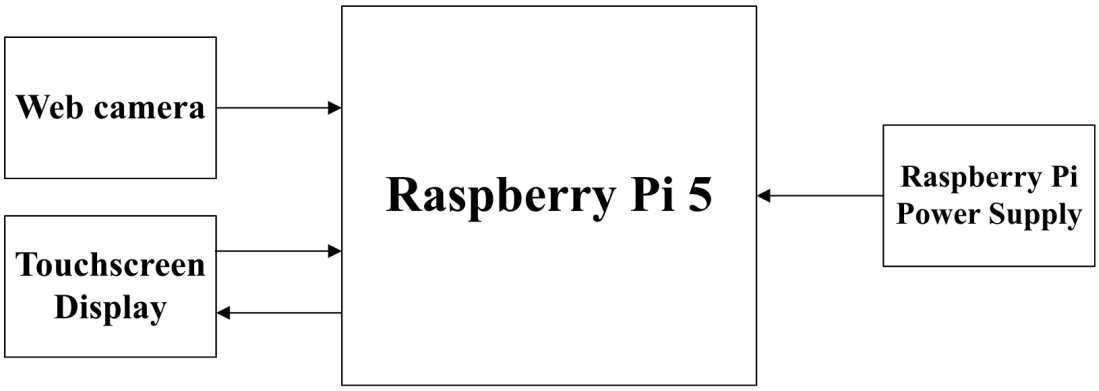


Figure 3.2: Hardware system block diagram

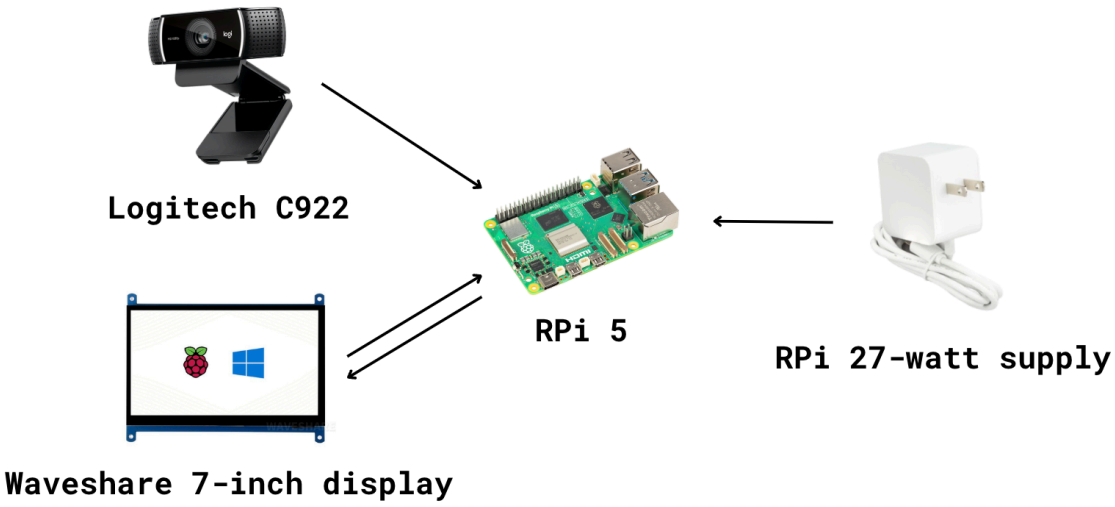


Figure 3.3: Graphic implementation diagram

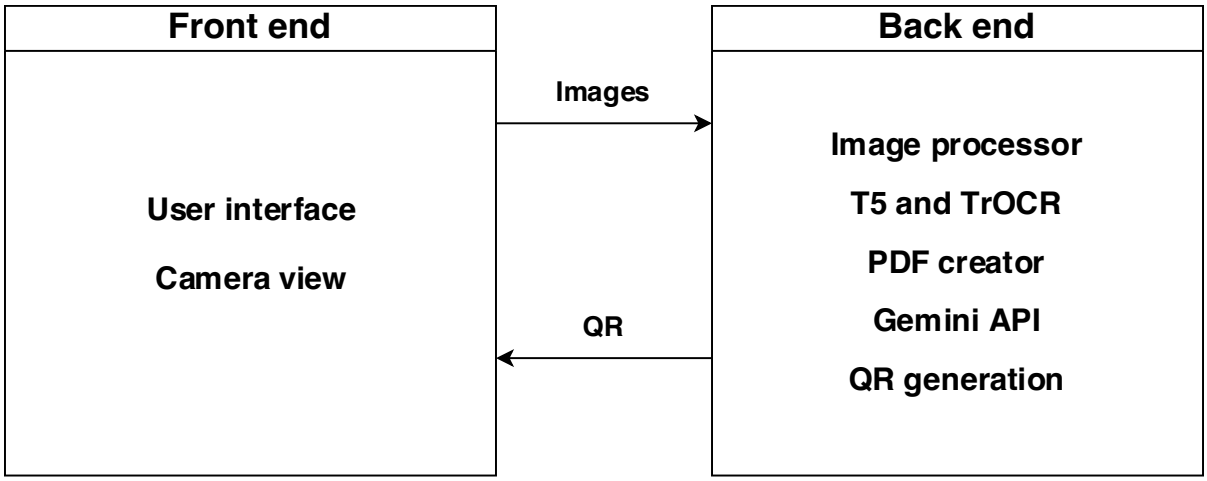


Figure 3.4: Program architecture

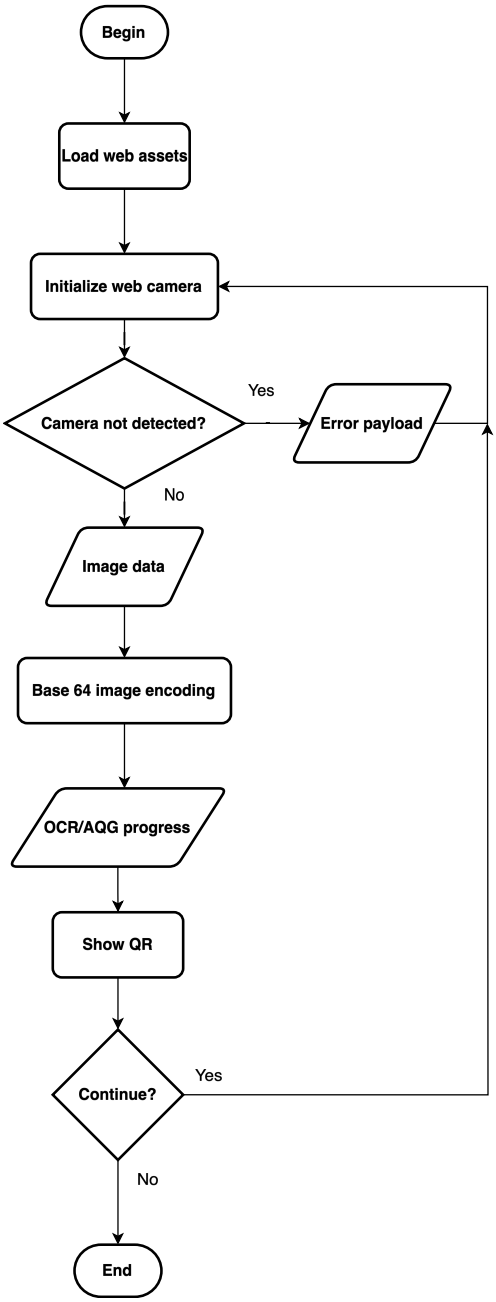


Figure 3.5: Front-end program flow

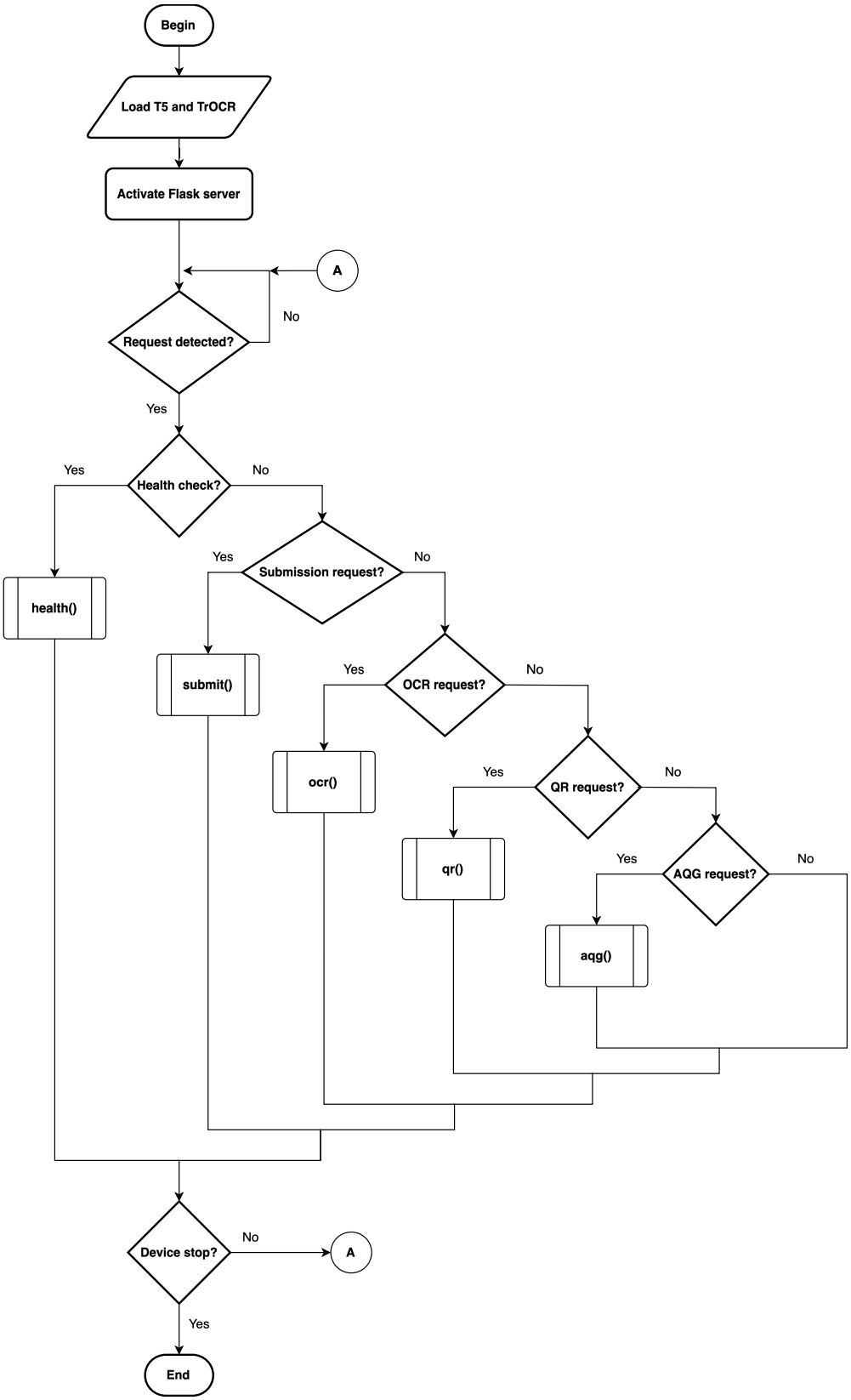


Figure 3.6: Back-end program flow

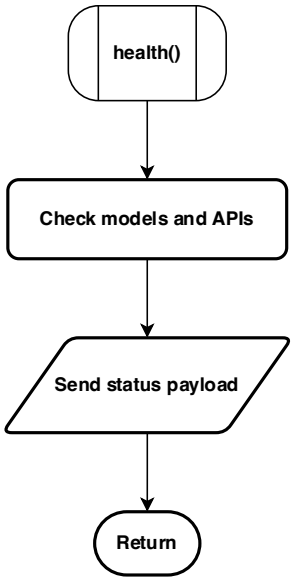


Figure 3.7: Health check subroutine flowchart

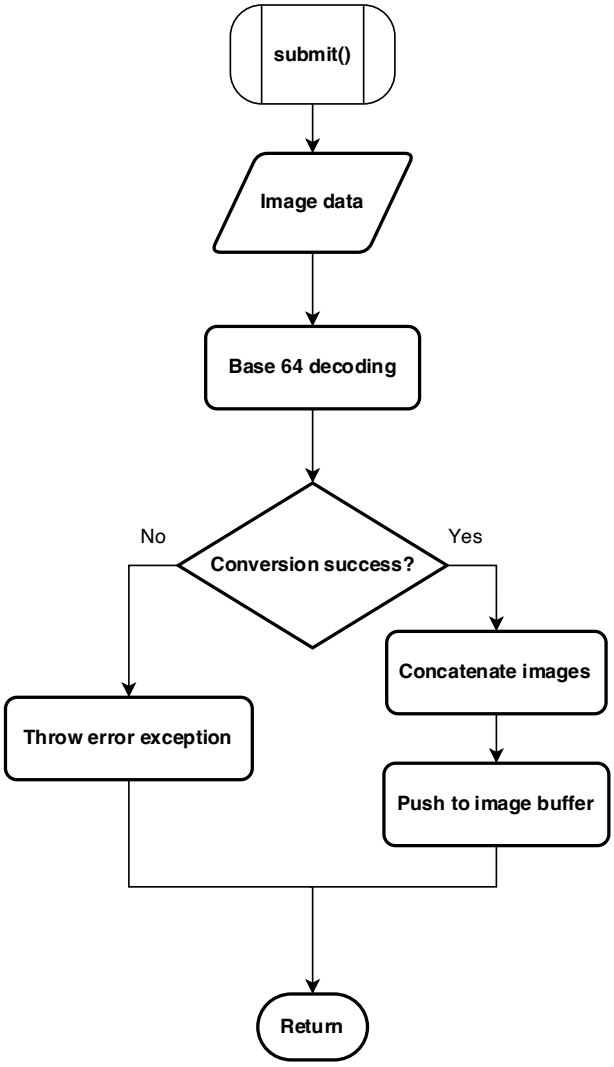


Figure 3.8: Submit subroutine flow

Figure 3.9: OCR subroutine flow

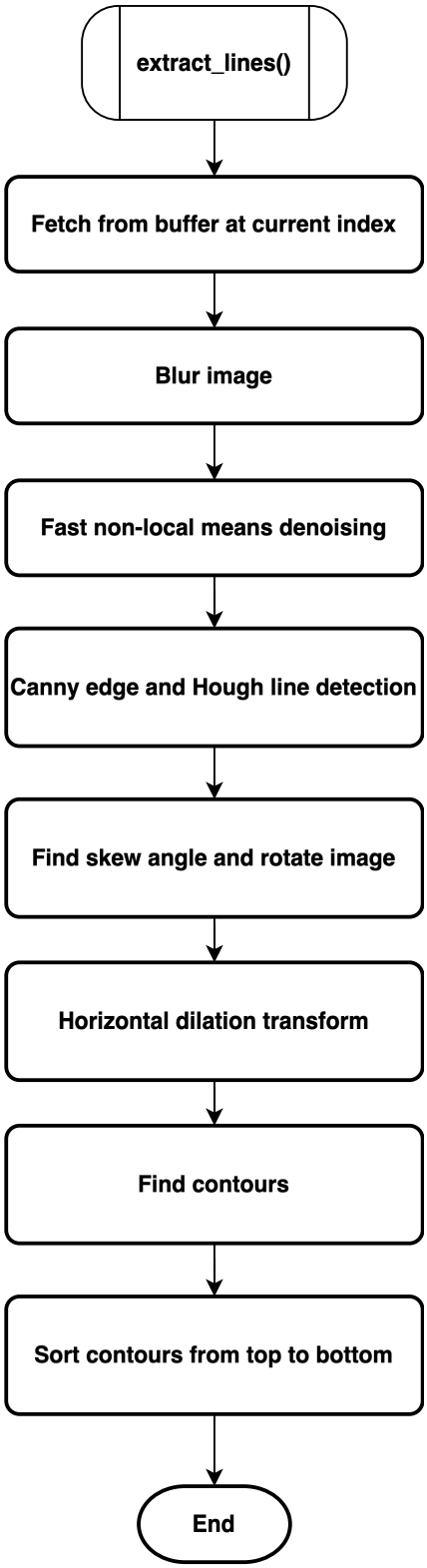


Figure 3.10: Line extraction subroutine flow

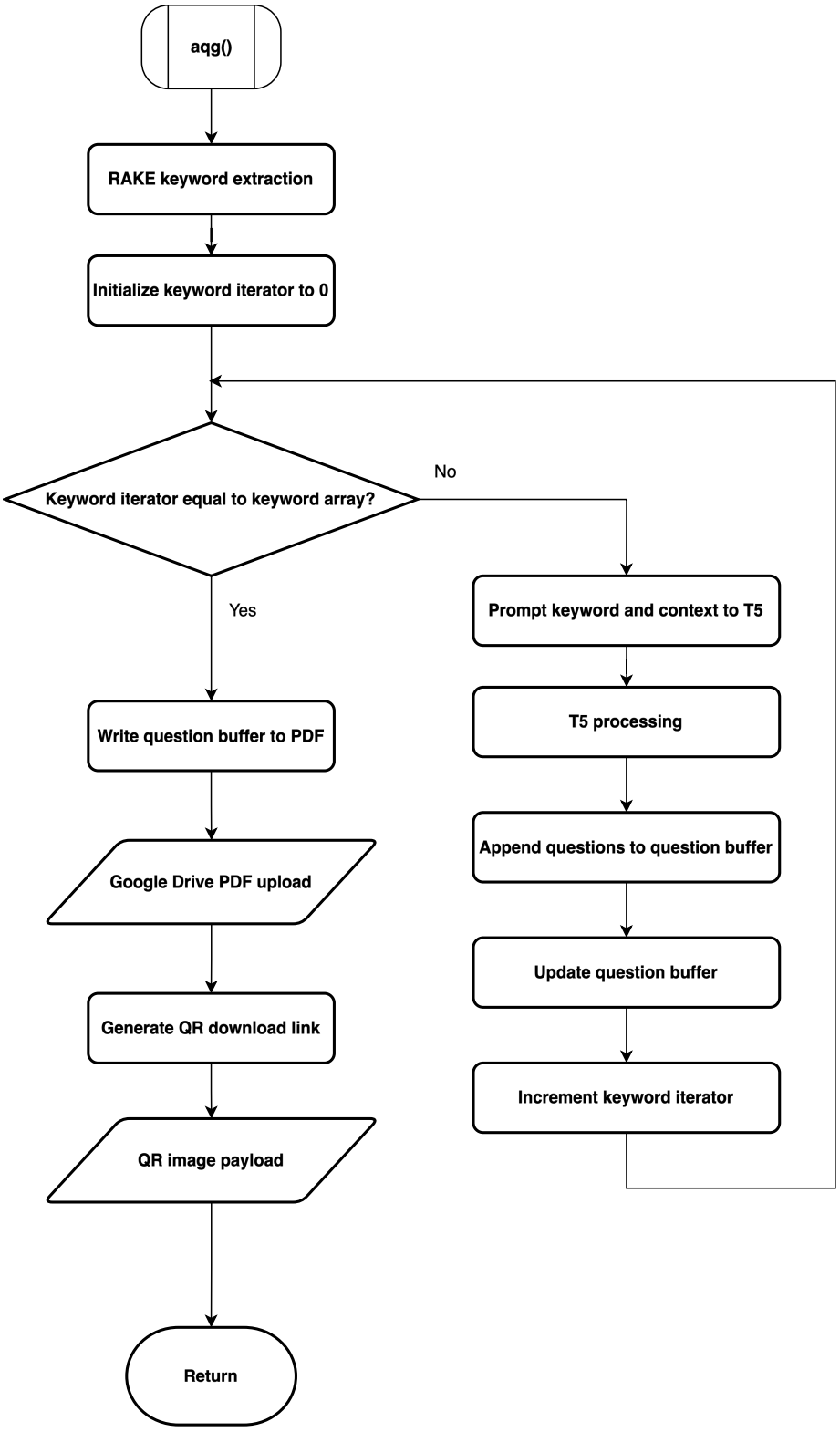


Figure 3.11: AQG subroutine flow

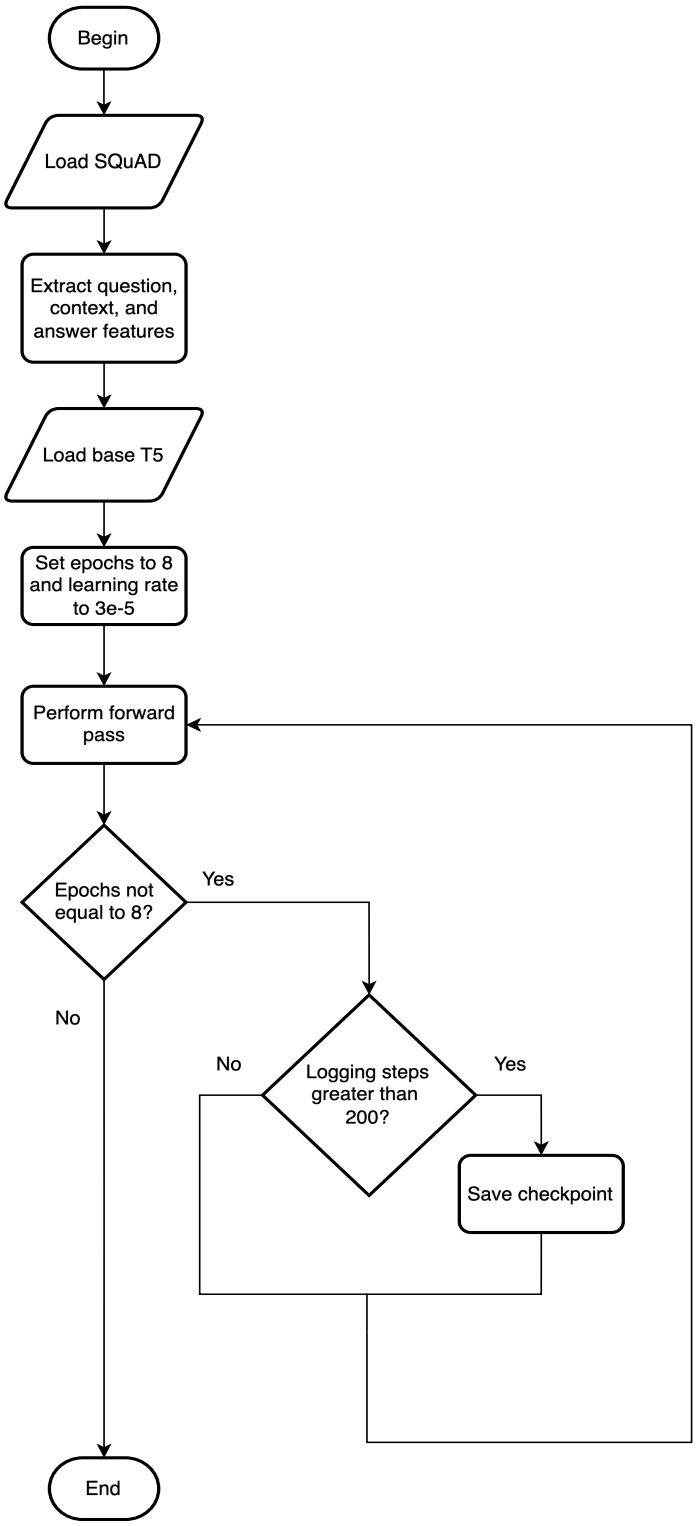


Figure 3.12: T5 fine tuning flow

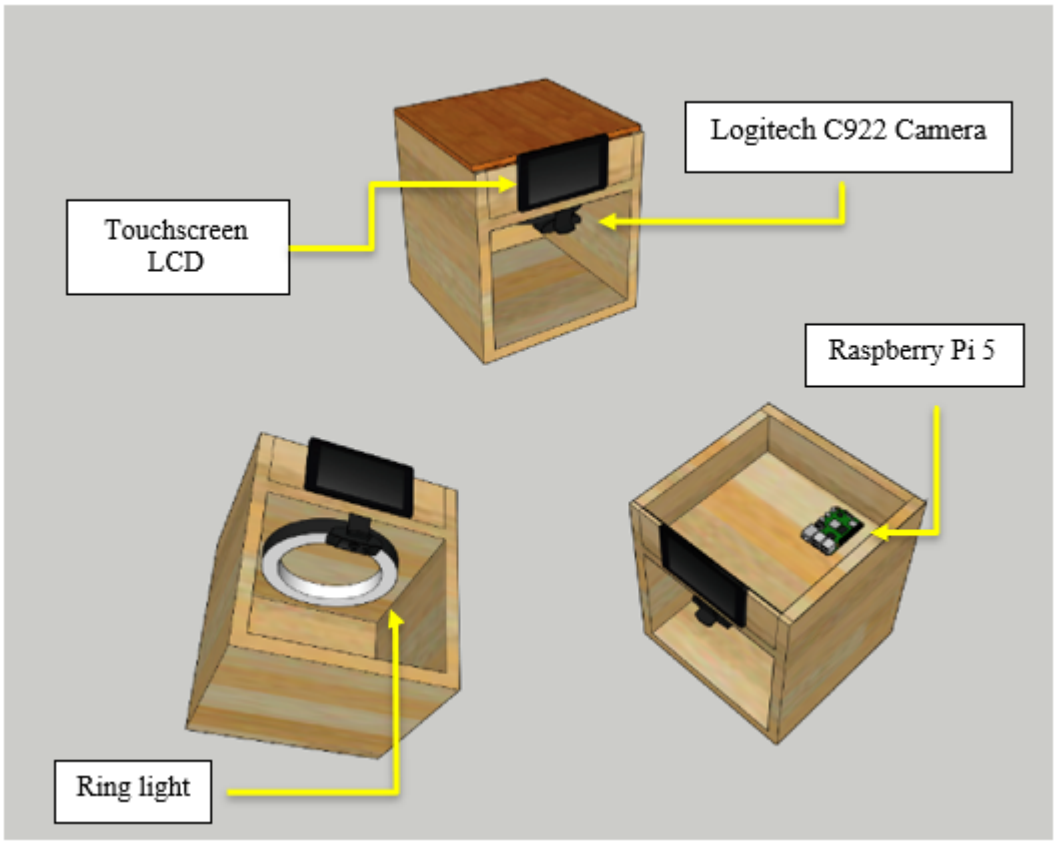


Figure 3.13: Experimental setup

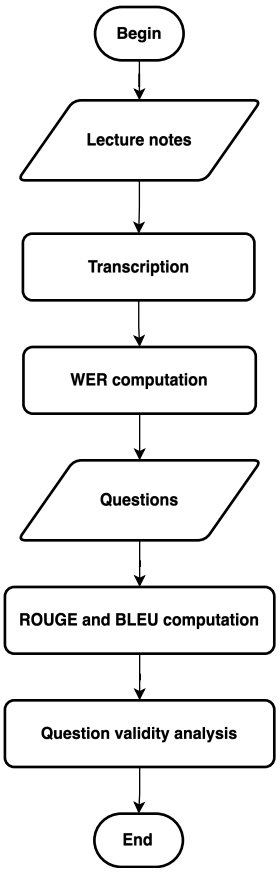


Figure 3.14: Data gathering procedure flowchart

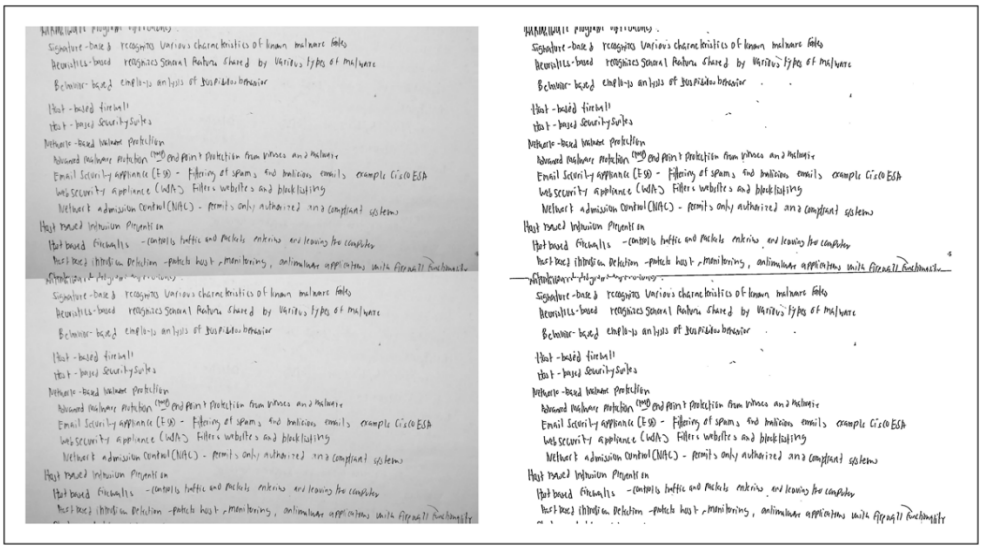


Figure 3.15: Sample handwritten note capture (left) and processed version (right)

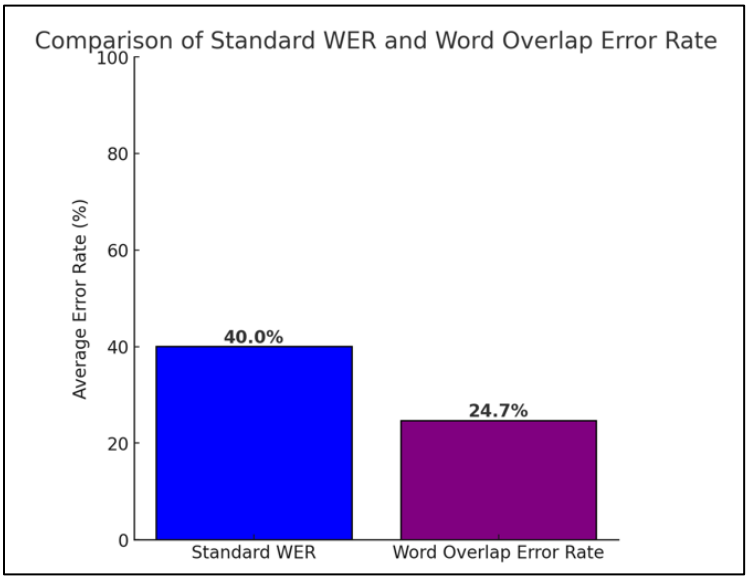


Figure 3.16: WER and word overlap error rate (WOER) histogram

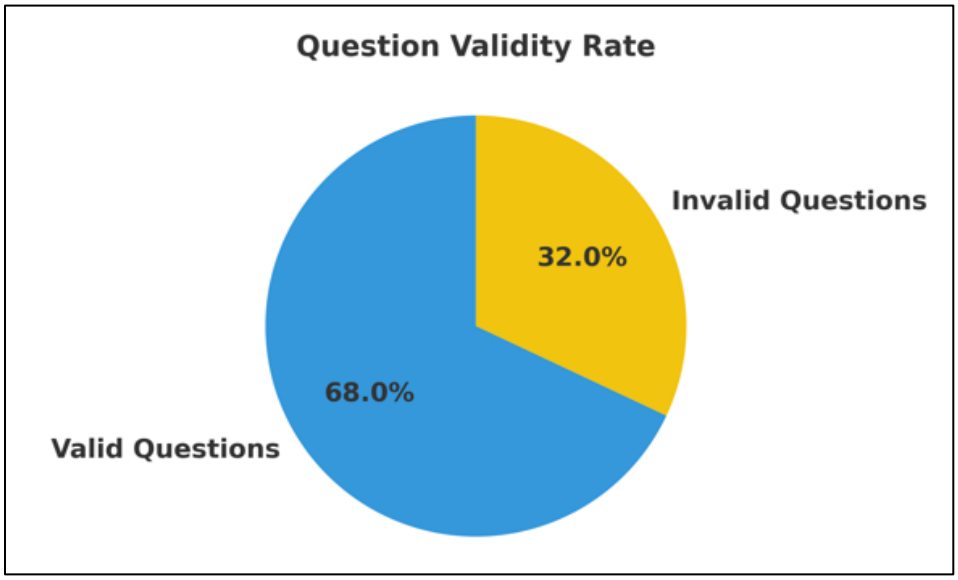


Figure 3.17: Question validity pie chart

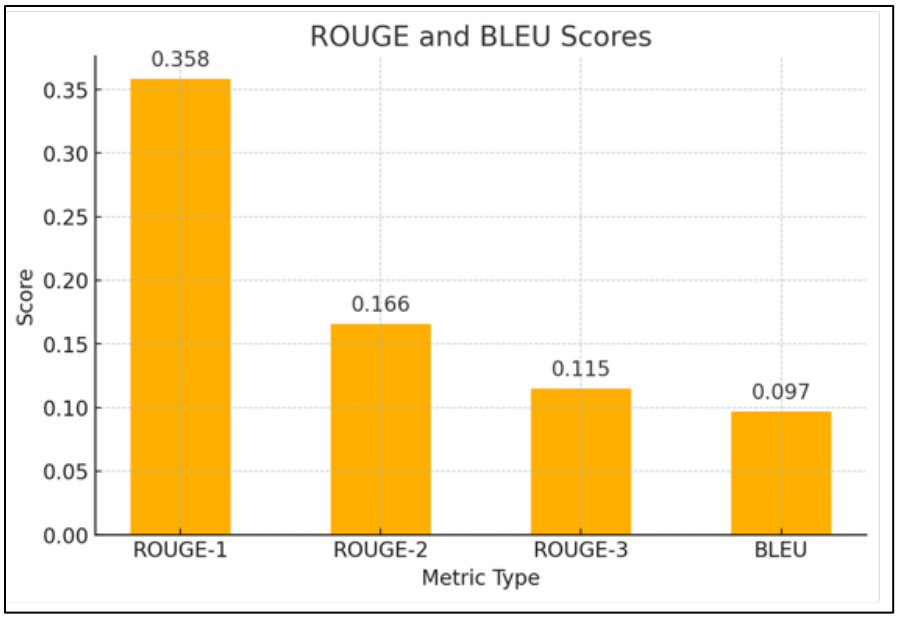


Figure 3.18: ROUGE and BLEU histogram

APPENDIX B: TABLES

Question Type		Example
Subjective Questions	Opinion-Based	Do you want to live on the moon?
Objective Questions	Factoid-Type	What color is the moon?
	List-Type	List all the moons of the solar system.
	Confirmation	Is the Sun a star?
	Causal	How do the planets orbit the Sun?
	Hypothetical	What if the Sun explodes?

Table 2.1: Question classification table (sourced from [5] **and** [6]**)**

ID	573610e1012e2f140011a182
Title	Hunting
Context	Hunting big game typically requires a "tag" for each animal harvested. Tags must be purchased in addition to the hunting license, and the number of tags issued to an individual is typically limited. In cases where there are more prospective hunters than the quota for that species, tags are usually assigned by lottery. Tags may be further restricted to a specific area, or wildlife management unit. Hunting migratory waterfowl requires a duck stamp from the Fish and Wildlife Service in addition to the appropriate state hunting license.
Question	What is required of hunting migratory waterfowl?
Answers	{ "text": ["duck stamp"], "answer_start": [439] }

Table 2.2: Sample data entry in SQuAD (sourced from [12]**)**

Index	**Handwritten Text**	
	Student	**Model**
1		
2		
…	…	…
69		
70		

Table 3.1: Testing table for OCR

Note index	Question index	Question
1	1	
	2	
	3	
	4	
	5	
…	…	
70	346	
	347	
	348	
	349	
	350	

Table 3.2: Testing table for question validity

Index	Questions	
	Student	Model
1		
…	…	…
135		

Table 3.3: Testing table for question comparison

Index	WER
1	
…	…
70	
Average	

Table 3.4: Evaluation table for OCR

Note index	Question index	Valid?
1	1	
	2	
	3	
	4	
	5	
…	…	…
70	346	
	347	
	348	
	349	
	350	
Total valid		

Table 3.5: Evaluation table for question validity

Index	ROUGE	BLEU
1		
…	…	…
70		
Average		

Table 3.6: Evaluation table for AQG

APPENDIX C: CODES

Back-end main loop

```
import cv2
import numpy as np
import base64
from pdf_generation import generate_numbered_pdf
from ocr import process_and_crop_image, batch_infer_trocr
from base64_processor import decode_base64_image
from aqg import aqg
from variables import global_state
from drive import upload_file_to_drive_and_get_qr_b64
from flask import Flask, request, jsonify
from flask_cors import CORS

app = Flask(__name__)
CORS(app, resources={r"/*": {"origins": "http://localhost:4200"}})

@app.route('/status', methods=['GET'])
def get_status():
    print(f"Status: {global_state['processed_count']}/{global_state['original_total']}")
    return jsonify({
        "currentStatus": global_state["processed_count"],
        "totalStatus": global_state["original_total"],
        "currentSubStatus": global_state["processed_subcount"],
        "totalSubStatus": global_state["total_subcount"],
        "phase": global_state["phase"]
    })

@app.route('/ocr', methods=['POST'])
def ocr_on_buffer():
    if not global_state["image_list"]:
        return jsonify({"status": "empty"})

    if global_state["original_total"] == 0:
        global_state["original_subtotal"] = 0
        global_state["processed_count"] = 0
        global_state["procssed_subcount"] = 0

    while global_state["image_list"]:
        global_state["original_total"] = len(global_state["image_list"])
        img = global_state["image_list"].pop(0)
        image_binary = base64.b64decode(img)
        image = cv2.imdecode(np.frombuffer(image_binary, np.uint8), cv2.IMREAD_GRAYSCALE)
        cropped_images_array = process_and_crop_image(image)
        corrected_text = batch_infer_trocr(cropped_images_array)
        global_state["text_list"].append(corrected_text)
        global_state["processed_count"] += 1
        print(f"Processed
{global_state['processed_count']}/{global_state['original_total']} images.")

    global_state["phase"] = 1
    global_state["original_total"] = len(global_state["text_list"])
    global_state["processed_count"] = 0
    global_state["subcount"] = 0

    while global_state["text_list"]:
        text = global_state["text_list"].pop(0)
        aqg_text = aqg(text)
        for a in aqg_text:
            global_state["aqg_list"].append(a)
        global_state["processed_count"] += 1

    generate_numbered_pdf(global_state["aqg_list"], "test.pdf")
    b64_qr = upload_file_to_drive_and_get_qr_b64("test.pdf", "test.pdf")
    global_state["image"] = b64_qr
    global_state["phase"] = 0
```

```
    global_state["original_total"] = 0
    global_state["processed_count"] = 0
    global_state["processed_subcount"] = 0
    global_state["total_subcount"] = 0
    global_state["aqg_list"] = []
    global_state["text_list"] = []

    return jsonify({"status": b64_qr})

@app.route('/qr', methods=['GET'])
def release_qr():
    return jsonify({'image': global_state["image"]})

@app.route('/submit', methods=['POST'])
def merge_images():
    try:
        data = request.get_json()
        image1_base64 = data.get('image1')
        image2_base64 = data.get('image2')
        if not image1_base64 or not image2_base64:
            return jsonify({"error": "Missing image data"}), 400
        top_image = decode_base64_image(image1_base64)
        bottom_image = decode_base64_image(image2_base64)
        if top_image is None or bottom_image is None:
            return jsonify({"error": "Invalid image data"}), 400
        if top_image.shape[1] != bottom_image.shape[1]:
            width = min(top_image.shape[1], bottom_image.shape[1])
            top_image = cv2.resize(top_image, (width, top_image.shape[0]))
            bottom_image = cv2.resize(bottom_image, (width, bottom_image.shape[0]))
        merged_image = cv2.vconcat([top_image, bottom_image])
        _, buffer = cv2.imencode('.jpg', merged_image)
        merged_image_base64 = base64.b64encode(buffer).decode('utf-8')
        global_state["image_list"].append(merged_image_base64)
        return jsonify({"status": "ok"}), 200
    except Exception as e:
        return jsonify({"error": str(e)}), 500

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=6969)
```

OCR function

```
from transformers import TrOCRProcessor, VisionEncoderDecoderModel
from PIL import Image
from tqdm import tqdm
from variables import global_state
import google.generativeai as genai
import torch
import cv2
import numpy as np

trocr_device = torch.device('cpu')
trocr_processor = TrOCRProcessor.from_pretrained('microsoft/trocr-small-handwritten',
use_fast=True)
trocr_model = VisionEncoderDecoderModel.from_pretrained('microsoft/trocr-small-
handwritten').to(trocr_device)

genai.configure(api_key=[redacted])
gemini_model = genai.GenerativeModel("gemini-1.5-flash")

def apply_scanner_effect(image):
    if len(image.shape) == 3:
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    else:
        gray = image
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)
    denoised = cv2.fastNlMeansDenoising(blurred, None, h=1, templateWindowSize=7,
searchWindowSize=21)
    scanned_denoised = cv2.adaptiveThreshold(
```

```
        denoised, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2
    )
    return scanned_denoised

def process_and_crop_image(image):
    edges = cv2.Canny(image, 50, 150, apertureSize=3)
    lines = cv2.HoughLines(edges, 1, np.pi / 180, 200)
    def fine_tuned_angle(lines):
        angles = []
        if lines is not None:
            for rho, theta in lines[:, 0]:
                if np.abs(theta - np.pi / 2) < np.deg2rad(20):
                    angle = (theta - np.pi / 2) * (180 / np.pi)
                    angles.append(angle)
        return np.median(angles) if angles else 0
    adjusted_angle = fine_tuned_angle(lines)
    (h, w) = image.shape[:2]
    center = (w // 2, h // 2)
    rotation_matrix_adjusted = cv2.getRotationMatrix2D(center, adjusted_angle, 1.0)
    adjusted_rotated_image = cv2.warpAffine(image, rotation_matrix_adjusted, (w, h),
                                            flags=cv2.INTER_LINEAR,
borderMode=cv2.BORDER_REPLICATE)
    cv2.imwrite("./debug/adjusted_rotated_image.png", adjusted_rotated_image)
    thresholded_image = apply_scanner_effect(adjusted_rotated_image)
    cv2.imwrite("thresholded.png", thresholded_image)
    _, binary = cv2.threshold(thresholded_image, 127, 255, cv2.THRESH_BINARY_INV)
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (100, 5))
    dilated = cv2.dilate(binary, kernel, iterations=1)
    cv2.imwrite("./debug/dilated.png", dilated)
    contours, _ = cv2.findContours(dilated, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    sorted_contours = sorted(contours, key=lambda cnt: cv2.boundingRect(cnt)[1])
    target = thresholded_image.copy()
    cropped_regions = []
    i = 0
    for cnt in sorted_contours:
        x, y, w, h = cv2.boundingRect(cnt)
        cropped_closed = target[y:y + h, x:x + w]
        if w >= 175 and h > 20:
            cv2.rectangle(
                target,
                (x, y),
                (x + w, y + h),
                (0, 255, 0),
                2
            )
            cv2.imwrite(f"test/{i}.png", cropped_closed)
            i += 1
            cropped_regions.append(cv2.merge([cropped_closed, cropped_closed,
cropped_closed]))
    cv2.imwrite("target.png", target)
    return cropped_regions

def batch_infer_trocr(cropped_images):
    global_state["processed_subcount"] = 0
    global_state["total_subcount"] = len(cropped_images)
    def extract_response(response):
      candidate = response.candidates[0]
      full_text = ""
      for part in candidate.content.parts:
        full_text += part.text
      return full_text
    recognized_texts = []
    for cropped_image in tqdm(cropped_images, desc="Processing images"):
        pil_image = Image.fromarray(cropped_image)
        pil_image = pil_image.convert("RGB")
        inputs = trocr_processor(pil_image, return_tensors="pt").to(trocr_device)
        outputs = trocr_model.generate(**inputs)
        recognized_text = trocr_processor.batch_decode(outputs,
skip_special_tokens=True)[0]
        recognized_texts.append(recognized_text)
        global_state["processed_subcount"] += 1
```

```
    combined_text = " ".join(recognized_texts)
    print(combined_text)
    gemini_response = gemini_model.generate_content(f"Refine the OCR-extracted text below
by correcting errors, removing any outlier/irrelevant text, and logically extrapolating
missing context where needed. Feel free to remove any outliers but ensure the output is
coherent and complete. Return only the enhanced text without explanations. TextRefine the
OCR-extracted text below by correcting errors, maintaining the original structure and
intent, and logically extrapolating missing context where needed. Prioritize textual
integrity but ensure the output is coherent and complete. Return only the enhanced text
without explanations. Text:  {combined_text}")
    corrected_text = extract_response(gemini_response)
    print(corrected_text)
    return corrected_text
```

AQG function

```
from transformers import T5Tokenizer, T5ForConditionalGeneration
from variables import global_state
import torch
import spacy
import nltk
from rake_nltk import Rake

nltk.download('stopwords')
nltk.download('punkt_tab')

def extract_top_5_keywords(context):
    rake = Rake()
    rake.extract_keywords_from_text(context)
    top_5_keywords = rake.get_ranked_phrases()[:5]
    return top_5_keywords

aqg_model =
T5ForConditionalGeneration.from_pretrained("./T5_model").to(torch.device('cpu'))
aqg_tokenizer = T5Tokenizer.from_pretrained("./T5_model")

def aqg(text):
    keyword_array = extract_top_5_keywords(text)
    global_state["processed_subcount"] = 0
    global_state["total_subcount"] = len(keyword_array)
    print(keyword_array)
    questions = []
    for k in keyword_array:
        inputs = f"generate question: Answer: {k} | Context: {text}"
        input_ids = aqg_tokenizer.encode(inputs, return_tensors="pt", max_length=512,
truncation=True).to('cpu')
        outputs = aqg_model.generate(input_ids, max_length=256, repetition_penalty=1.2,
top_k=50, do_sample=True, top_p=0.90, temperature=0.9)
        decoded = aqg_tokenizer.decode(outputs[0], skip_special_tokens=True)
        print(decoded)
        questions.append(f"{decoded} hint: {k}")
        global_state["processed_subcount"] += 1
    return questions
```

Drive function

```
from google.oauth2 import service_account
from googleapiclient.discovery import build
from googleapiclient.http import MediaFileUpload
import os
import qrcode
import base64
from io import BytesIO

GOOGLE_DRIVE_FOLDER_ID = "[redact]"
GOOGLE_DRIVE_CREDENTIALS_PATH = os.path.join(os.getcwd(), "[redact]")

def authenticate_google_drive():
```

```
    creds = service_account.Credentials.from_service_account_file(
        GOOGLE_DRIVE_CREDENTIALS_PATH, scopes=["https://www.googleapis.com/auth/drive"]
    )
    return build("drive", "v3", credentials=creds)

def find_existing_file(drive_service, filename):
    query = f"'{GOOGLE_DRIVE_FOLDER_ID}' in parents and name = '{filename}'"
    results = drive_service.files().list(q=query, fields="files(id)").execute()
    files = results.get("files", [])
    return files[0]["id"] if files else None

def generate_qr_code_base64(data):
    qr = qrcode.QRCode(version=1, error_correction=qrcode.constants.ERROR_CORRECT_L,
box_size=10, border=4)
    qr.add_data(data)
    qr.make(fit=True)
    img = qr.make_image(fill="black", back_color="white")
    buffered = BytesIO()
    img.save(buffered, format="PNG")
    return base64.b64encode(buffered.getvalue()).decode("utf-8")

def upload_file_to_drive_and_get_qr_b64(filename, filepath):
    drive_service = authenticate_google_drive()
    existing_file_id = find_existing_file(drive_service, filename)
    if existing_file_id:
        print(f"Overwriting file: {filename} (ID: {existing_file_id})")
        file_metadata = {"name": filename}
        media = MediaFileUpload(filepath, mimetype="application/pdf", resumable=True)
        updated_file = drive_service.files().update(
            fileId=existing_file_id, body=file_metadata, media_body=media, fields="id"
        ).execute()
        file_id = updated_file.get("id")
    else:
        print(f"Uploading new file: {filename} to Google Drive...")
        file_metadata = {"name": filename, "parents": [GOOGLE_DRIVE_FOLDER_ID]}
        media = MediaFileUpload(filepath, mimetype="application/pdf", resumable=True)
        uploaded_file = drive_service.files().create(
            body=file_metadata, media_body=media, fields="id"
        ).execute()
        file_id = uploaded_file.get("id")
    drive_service.permissions().create(
        fileId=file_id, body={"role": "reader", "type": "anyone"}, fields="id"
    ).execute()
    direct_download_link = f"https://drive.google.com/uc?export=download&id={file_id}"
    print(f"File uploaded successfully: {direct_download_link}")
    return generate_qr_code_base64(direct_download_link)
```

B64 decoder function

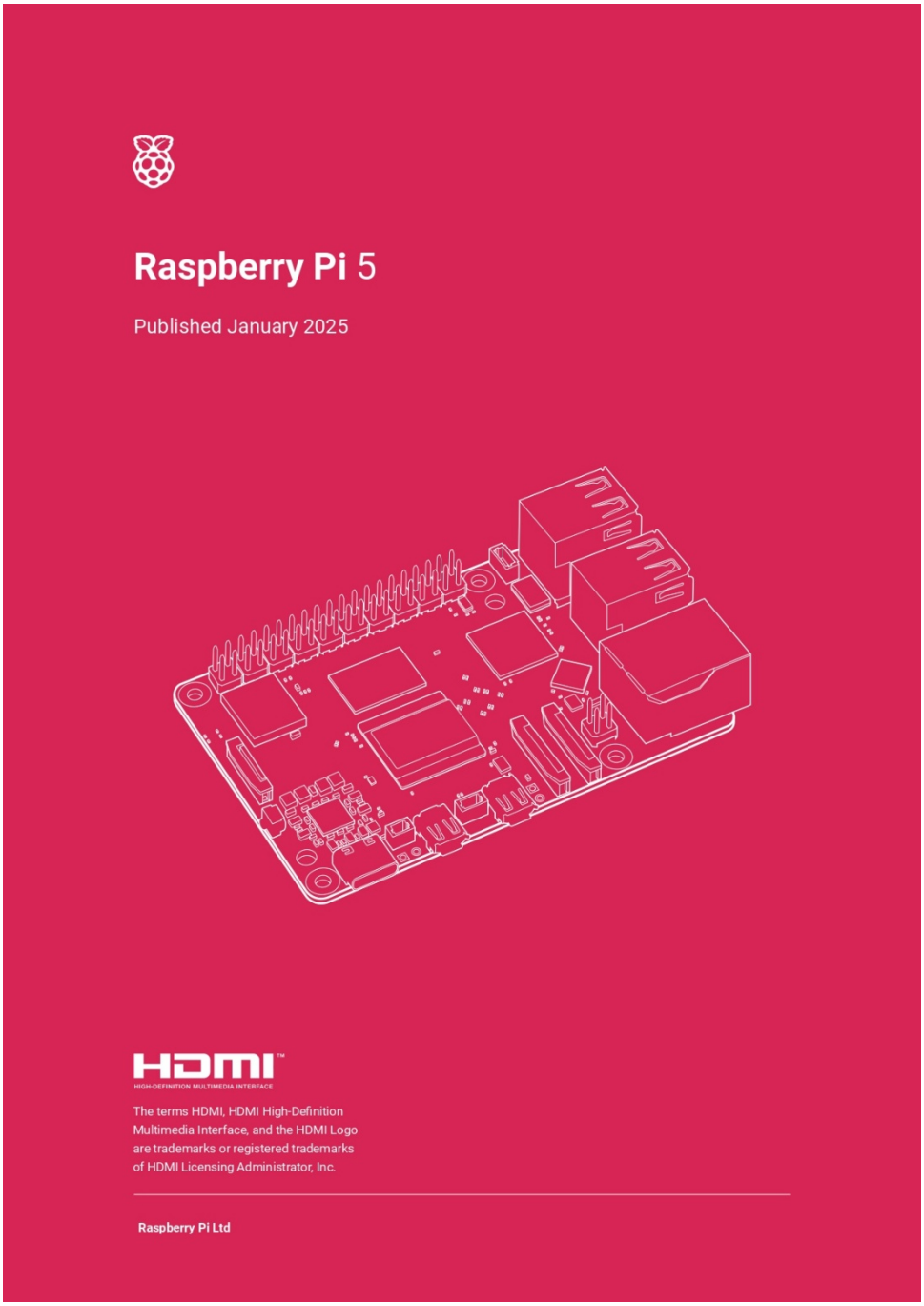
```
import base64
import cv2
import numpy as np

def decode_base64_image(base64_string):
    try:
        image_data = base64.b64decode(base64_string)
        np_arr = np.frombuffer(image_data, np.uint8)
        return cv2.imdecode(np_arr, cv2.IMREAD_COLOR)
    except Exception as e:
        return None

    return generate_qr_code_base64(direct_download_link)
```

APPENDIX D: DATASHEETS

Raspberry Pi 5



Overview

Welcome to the latest generation of Raspberry Pi: the everything computer.

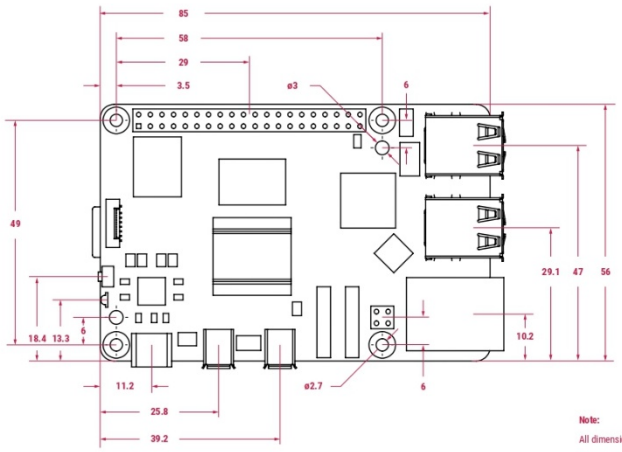
Featuring a 64-bit quad-core Arm Cortex-A76 processor running at 2.4GHz, Raspberry Pi 5 delivers a 2–3× increase in CPU performance relative to Raspberry Pi 4. Alongside a substantial uplift in graphics performance from an 800MHz VideoCore VII GPU; dual 4Kp60 display output over HDMI; and state-of-the-art camera support from a rearchitected Raspberry Pi Image Signal Processor, it provides a smooth desktop experience for consumers, and opens the door to new applications for industrial customers.

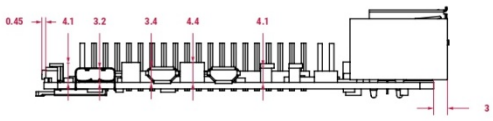
For the first time, this is a full-size Raspberry Pi computer using silicon built in-house at Raspberry Pi. The RP1 "southbridge" provides the bulk of the I/O capabilities for Raspberry Pi 5, and delivers a step change in peripheral performance and functionality. Aggregate USB bandwidth is more than doubled, yielding faster transfer speeds to external UAS drives and other high-speed peripherals; the dedicated two-lane 1Gbps MIPI camera and display interfaces present on earlier models have been replaced by a pair of four-lane 1.5Gbps MIPI transceivers, tripling total bandwidth, and supporting any combination of up to two cameras or displays; peak SD card performance is doubled, through support for the SDR104 high-speed mode; and for the first time the platform exposes a single-lane PCI Express 2.0 interface, providing support for high-bandwidth peripherals.

Specification

Processor	Broadcom BCM2712 2.4GHz quad-core 64-bit Arm Cortex-A76 CPU, with Cryptographic Extension, 512KB per-core L2 caches, and a 2MB shared L3 cache
Features:	• VideoCore VII GPU, supporting OpenGL ES 3.1, Vulkan 1.2 • Dual 4Kp60 HDMI® display output with HDR support • 4Kp60 HEVC decoder • LPDDR4X-4267 SDRAM (options for 2GB, 4GB, 8GB and 16GB) • Dual-band 802.11ac Wi-Fi® • Bluetooth 5.0 / Bluetooth Low Energy (BLE) • microSD card slot, with support for high-speed SDR104 mode • 2 × USB 3.0 ports, supporting simultaneous 5Gbps operation • 2 × USB 2.0 ports • Gigabit Ethernet, with PoE+ support (requires separate PoE+ HAT) • 2 × 4-lane MIPI camera/display transceivers • PCIe 2.0 x1 interface for fast peripherals (requires separate M.2 HAT or other adapter) • 5V/5A DC power via USB-C, with Power Delivery support • Raspberry Pi standard 40-pin header • Real-time clock (RTC), powered from external battery • Power button
Production lifetime:	Raspberry Pi 5 will remain in production until at least January 2036
Compliance:	For a full list of local and regional product approvals, please visit pip.raspberrypi.com
List price:	2GB **$50** 4GB **$60** 8GB **$80** 16GB **$120**

Physical specification





Note:

All dimensions in mm

All dimensions are approximate and for reference purposes only. The dimensions shown should not be used for producing production data

The dimensions are subject to part and manufacturing tolerances

Not all of the board components are shown. Please reference a physical board for representation of componentry

Dimensions may be subject to change

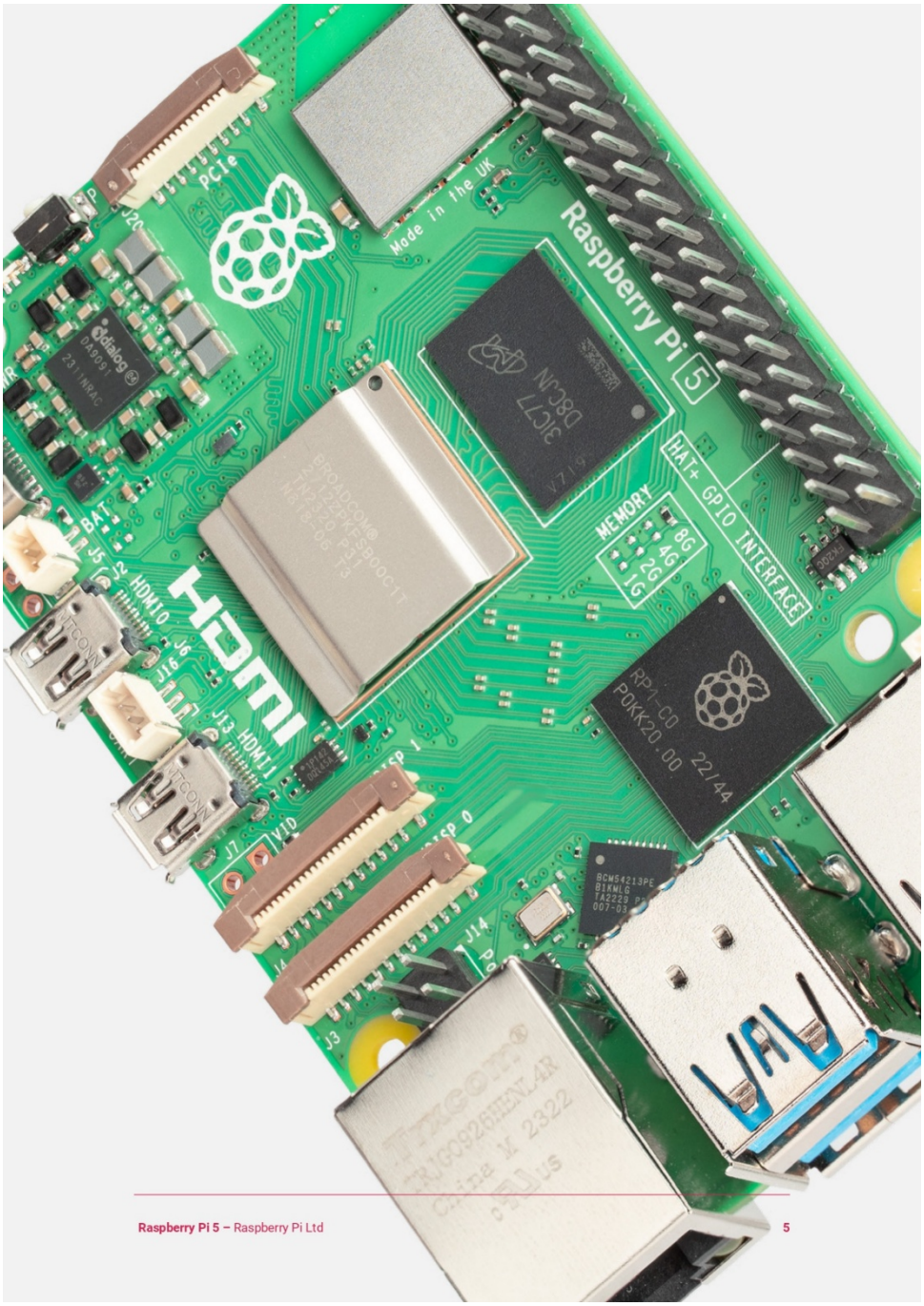
WARNINGS

- This product should be operated in a well ventilated environment, and if used inside a case, the case should not be covered.
- While in use, this product should be firmly secured or should be placed on a stable, flat, non-conductive surface, and should not be contacted by conductive items.
- The connection of incompatible devices to Raspberry Pi 5 may affect compliance, result in damage to the unit, and invalidate the warranty.
- All peripherals used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met.

SAFETY INSTRUCTIONS

To avoid malfunction or damage to this product, please observe the following:

- Do not expose to water or moisture, or place on a conductive surface while in operation.
- Do not expose to heat from any source; Raspberry Pi 5 is designed for reliable operation at normal ambient temperatures.
- Store in a cool, dry location.
- Take care while handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- While it is powered, avoid handling the printed circuit board, or handle it only by the edges, to minimise the risk of electrostatic discharge damage.

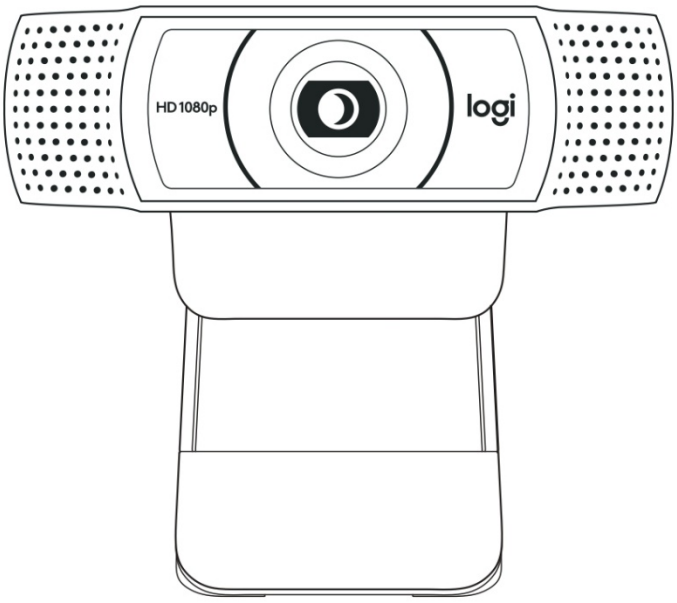

Raspberry Pi 5
Made in the UK
PCIe
HDMI
HDMI0
HDMI1
BAT
MEMORY
8G
4G
2G
1G
HAT+ GPIO INTERFACE
RP1-C0
POKK20.00
22/44
BCM54213PE
Raspberry Pi 5 – Raspberry Pi Ltd
5


Raspberry Pi
Raspberry Pi is a trademark of Raspberry Pi Ltd

Logitech C922

C922 PRO HD STREAM WEBCAM

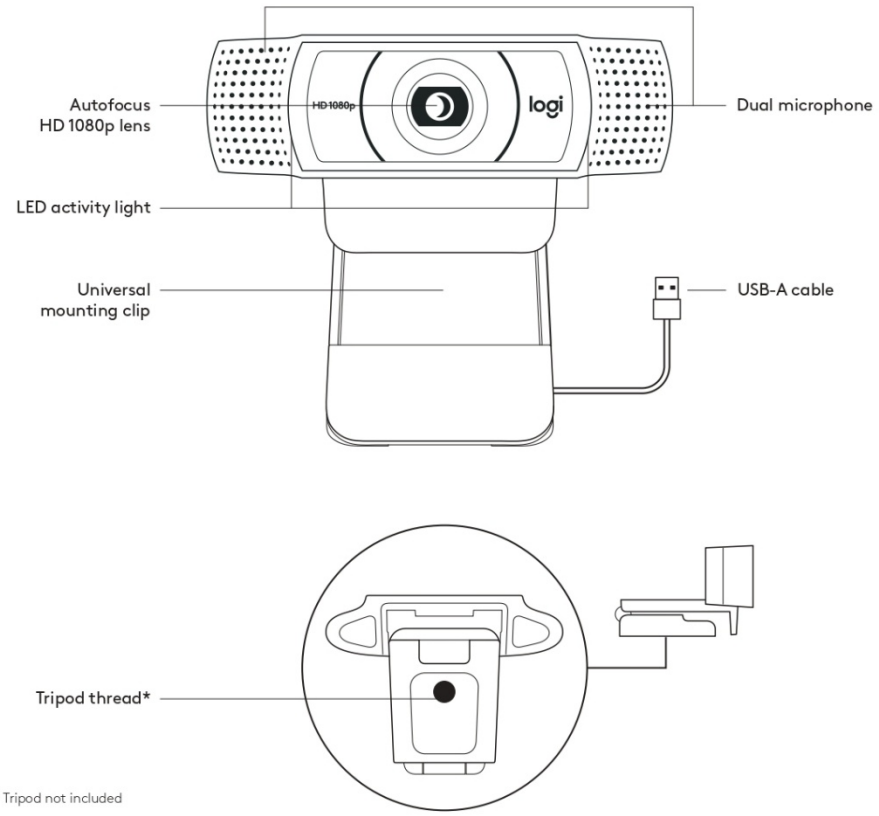
Complete Setup Guide
Guide d'installation complet



logitech

CONTENTS

KNOW YOUR PRODUCT



* Tripod not included

WHAT'S IN THE BOX

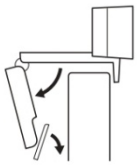
1 Webcam with 5 ft (1.5 m) attached USB-A cable

2 User documentation

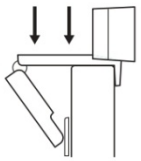
SETTING UP THE WEBCAM

For placement on a monitor

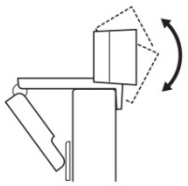
1 Place your webcam on a computer, laptop or monitor at a position or angle you desire.



2 Adjust the webcam to make sure the foot on the universal mounting clip is flush with the back of your device.

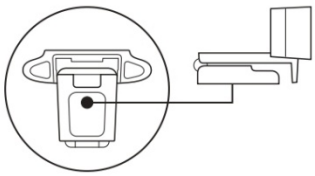


3 Manually adjust the webcam up/down to the best position to frame yourself.

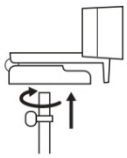


For placement on a tripod*

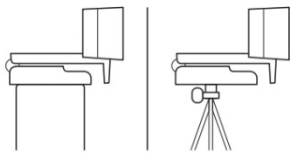
1 Locate the ¼ inch tripod thread on the bottom of the universal mounting clip.



2 Secure the webcam on your tripod by twisting it into the ¼ inch thread.



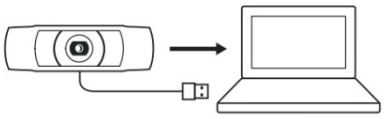
3 Place your webcam with the tripod anywhere you desire to the best position to frame yourself.



* Tripod not included

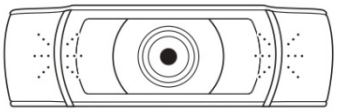
CONNECTING THE WEBCAM VIA USB-A

Plug the USB-A connector into the USB-A port on your computer.



SUCCESSFUL CONNECTION

LED activity light will light up when the webcam is in use by an application.



DIMENSIONS

INCLUDING FIXED MOUNTING CLIP:

Height x Width x Depth:
1.73 in (44 mm) x 3.74 in (95 mm) x 2.80 in (71 mm)

Cable Length: 5 ft (1.5 m)

Weight: 5.71 oz (162 g)

www.logitech.com/support/C922

MAPÚA UNIVERSITY	**PUBLISHING AGREEMENT FORM (Copyright and Consent Form)**	Document No: FM-EEC-19-00
		Effective Date: Nov 5, 2018

A. DECLARATION

TITLE OF ARTICLE: Automatic Question Generation From Handwritten Lecture Notes Using TrOCR Text Recognition and T5 Language Processing

AUTHOR/S: Rommel John H. Ronduen, Jan Adrian C. Manzanero, and Analyn N. Yumang

TO BE PUBLISHED IN: _____

Please mark all that apply:

1. Relationship with MAPUA UNIVERSITY

   - ☑ Student (Undergraduate, MS, Phd)
   - ☐ Faculty
   - ☐ Others (Please specify) _____

2. Status as Author

   - ☐ Sole Author
   - ☑ One Author Signing on behalf of all Co-Authors
     (Please attach written authorization from co-authors)
   - ☐ Authorized representative of employer as the Article is a "work for hire".

A. AGREEMENT

This is a publication agreement and copyright license ("Agreement") regarding the above declared written article or work (Article) to be published by Analyn N. Yumang in _____.

The parties to this Agreement are Name of the Author or Co-author: Rommel John H. Ronduen Jan Adrian C. Manzanero ("Author")

and MAPUA UNIVERSITY ("Publisher").

MAPÚA UNIVERSITY	**PUBLISHING AGREEMENT FORM (Copyright and Consent Form)**	Document No: FM-EEC-19-00
		Effective Date: Nov 5, 2018

The Author hereby declares and warrants:

1. The Author has created and owns the copyright to the Article.
2. The Article submitted by the Author for review is original and was written by the above declared Author/s
3. The Article contains no libelous, unlawful statements or materials that violate the personal and proprietary rights of any person or entity.
4. The Author has the full power and authority to enter into this Agreement and to grant the rights granted in this Agreement. The Author represents and warrants that the Article furnished to the Publisher has not been published previously. For purposes of this paragraph, making a copy of the Article accessible over the Internet, including, but not limited to, posting the Article to a database accessible over the Internet, does not constitute prior publication so long as such copy indicates that the Article is not in final form, such as by designating such copy to be a "draft," a "working paper," or "work-in-progress".
5. The Author grants to the Publisher a royalty-free, irrevocable, perpetual, worldwide nonexclusive license to publish, reproduce, display, distribute, and use the Article in any form, either separately or as part of a collective work, including but not limited to a nonexclusive license to publish the Article in the Journal, its electronic archive, its paper archive, or any collection of the Journal's works in any form whatsoever, copy and distribute individual reprints of the Article, and authorize reproduction and distribution of the Article or an abstract thereof by means of computerized retrieval systems.
6. The Author retains ownership of all rights under copyright in the Article, and all rights not expressly granted in this Agreement.
7. The Author agrees that s/he shall not publish the Article in any other journal or edited collection, whether electronic or otherwise, in substantially the same form as the Article, without acknowledging prior publication in the Journal with the sentence "This article has previously been published in _____", specifying the journal, book or manual where it was previously published.
8. The Author has obtained written permission from copyright owners from copyrighted works that are included and have properly credited the sources in the Article.
9. The Article is not subject to any prior rights or licenses, except those expressly authorized under this Agreement, and if any of the Author or co-Author/s institution has a policy that might restrict the author's ability to grant rights under this Agreement, a written waiver of that policy has been obtained.
10. This Agreement shall become effective and binding at the date of formal acceptance of the Article for publication by the Journal.

I HAVE READ AND AGREE FULLY WITH THE TERMS OF THIS AGREEMENT.

AUTHOR
Signed: Analyn N. Yumang 06/07/2025, Rommel John H. Ronduen 06/09/25, Jan Adrian C. Manzanero 06/09/25 Date: 06/07/2025

PUBLISHER
Signed: _____