VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
HO CHI MINH UNIVERSITY OF TECHNOLOGY
COMPUTER SCIENCE AND ENGINEERING FACULTY

———————— * ————————

THESIS PROPOSAL

Name of THESIS

Committee: Computer Science
Advisor: Nguyễn An Khương
Reviewer: No name

—o0o—

Students:
Nguyễn Nguyên Phương 1712726
Nguyễn Đình Thắng 1752048

HO CHI MINH CITY, 12/2020

Declaration

We hereby undertake that this is our own research project under the guidance of Dr. Nguyen. Research content and results are truthful and have never been published before. The data used for the analysis and comments are collected by us from many different sources and will be clearly stated in the references.

In addition, we also use a number of reviews and figures of other authors and organizations. All have citations and origins.

If we detect any fraud, we take full responsibility for the content of our graduate internship. Ho Chi Minh City University of Technology is not related to the copyright and copyright infringement caused by us in the implementation process.

Acknowledgments

First and foremost, we would like to express our sincere gratitude to our advisor Dr. Nguyen for the support of our thesis proposal for his patience, enthusiasm, experience and knowledge. He shared his experience and knowledge which helps us in our research and how to provide a good thesis proposal.

Abstract

Charts have been and always will be one of the most effective tools for demonstrating and sharing ideas among others. Besides text and images, drawing flow charts is the best way to give others a clearer path of the plan with the least amount of work. Nowadays, many meetings require a blackboard so everyone can express their thoughts on. This raised a problem with saving these drawings as a reference for future use since taking a picture of them will not solve the problem of re-editing these ideas and they need to be re-drawn to be suitable in professional documents. On the other hand, in order to digitalize the chart required to re-draw it using a computer or a special device like drawing boards and digital pens, which cost a lot and is not the most convenient tools to use.

Therefore, it is necessary to find a new way to convert the current hand-drawing charts into digital ones effortless, simplify the sharing process between users and be able to export them into another form like picture files (png, jpg), document files (pdf) or common diagram editing files (drawio). The application must be able to run on popular platforms and accessible by everyone.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Research in image recognition has a long journey. Within the image area, one smaller section deals with the detection of diagrams such as flowcharts, UML,... Since the introduction of the Online Handwritten Flowchart Dataset (OHFCD) in 2011, there has been numerous attempts in flowchart recognition. Flowchart recognition consists of two main approaches: online and offline recognition. In online recognition, the diagram is drawn as a sequence of strokes using a device with a touchscreen such as tablet and a pen or finger. In offline recognition, the diagram is a raw image from a source of image capture like phone camera. A few years ago, there has been more attention in online recognition as it reaches higher accuracy than offline recognition. However, when it comes to a pre-drawn diagram, where individual strokes cannot be captured, offline flowchart is preferred, though online recognition can still be constructed. For recognizing objects within images, object detectors based on Convolutional Neural Networks (CNN) are very common. While they can be applied to detect the individual symbols of a flowchart, an off-the-shelf object detector cannot be used to detect the relationships between elements of a graphic.

In captured image, there are many features that negatively affect the recognition process. Some other features are not needed in the process and may cause more problems when implementing the recognition algorithm, for example, color feature of the image is not a necessary feature in flowchart recognition which should be removed to reduce the work in the post-processing. Additionally, image distortion is the most pleasant problems in image processing. Image is distorted due to various type of noise such as Gaussian noise, Speckle noise, Salt and Pepper noise and many more other type of noise [1]. Noise is always appears in digital images during image acquisition, coding, transmission, and processing steps. It may arise in the image as effects of basic physics-like photon nature of light or thermal energy of heat inside the image sensors [2]. Noise tells unwanted information in digital image. Noise produces undesirable effects such as artifacts, unrealistic edges, unseen lines, corners, blurred objects and disturbs background scenes. Therefore, to achieve a good recognition result, pre-processing also plays an important role in the recognition process. Image pre-processing target is an improvement of the image quality that suppresses undesired distortions or enhances some image features relevant for further processing and analysis task. This process includes restoration state, whose aim is reducing or removed noise, negative impacts from environment condition in the image, and enhancement stage, whose goals are enhancing the main features of the image which make the recognition process work more effectively. In addition, some features of the image, which may be unnecessary according to the target of the recognition result, can be removed in pre-

processing to reduce the complexity of the recognition algorithm. There are many well-known techniques used in image processing such as grayscale image for color reduction, histogram equalization for image contrast adjustment, noise filter techniques such as Gaussian Blur, Median Blur. In addition, image is able to be converted into binary image, which only includes two colors black and white. There are also the techniques called morphological transformation which is normally used on binary image and can be used to enhance image features. However, these techniques has their own advantages and disadvantages, which needs to be carefully selected to give a acceptable image quality for the recognition process.

This project will not stop with the theory, but also apply it to build a complete product that serves actual clients to solve real-life problems. Nowadays, many meetings require to have a black board so people can express their ideas on. However, there is still a limited option when it takes to saving these sketches and transform them into digital form for storing and referencing in the future. Realizing the need of a product that saving these drawing, we decided to build an application that can convert hand-drawn ideas into digital form. This app should also have login system so that users can save their own data safely in the database and sharing them with their co-worker.

The report is organized as follows Chapter **??** briefly surveys application that can detect object and related work in diagram detection in general and flowchart detection in particular. Chapter 3 provides sufficient knowledge in order to implement the project. Chapter 4 shows our proposed system, including how the application works. Chapter 5 shows the implementation of the application and server. Chapter 6 lists our experiment and implementation of the system. Finally, Chapter 7 shows our challenge and potential future in the thesis.

1.2 Project Goals

The target of this project is to build a system that is able to convert hand drawn flowchart image into digital version and allows user to modify the result into the final product before sharing or converting it into other form.

Chapter 2

Related works

2.1 Recognition Application

There has been numerous application that can do certain object detection. Google Lens is an AI-powered application, designed to bring up relevant information related to the object. For example, when pointing the camera at a puppy, the application will be able to detect it and give related information, such as breeds, approximate age and food type. It uses Region Proposal Network (RPN) to detect character box for OCR and Convolutional Neural Network (CNN) with a Long short-term memory (LSTM) to detect object in image. Released in December 2019, Microsoft Math Solver application can recognize letters, math characters and symbols using OCR, categorizing using a natural language processing algorithm and then solved step by step. There are also multiple candidates that can detect certain type of object such as Aipoly Vision, ScreenShop, Flow and Vivino.

2.2 Object detection methods

2.2.1 Introduction

Object detection is a critical task in computer vision field that deal with multiple types of a certain visual object such as animal, flower, people, diagram,... It acts as the fundamental of other tasks such as segmentation, object tracking, image captioning, event detection, scene understanding and activity recognition,... The goal of object detection is to give basic knowledge to the computer: ”What is this object?” and if possible, return the coordinate where this object locates . The object detection task can be divided into two topics: ”general detection” and ”specific detection”. The first topic indicates the ability to detect instances of an object under an uniform condition and thus simulate human eyes with automotive car as an example. The other topic covers the detection of object under specific scenarios such as face detection, voice detection, text recognition, diagram recognition, etc. Object detection is widely used in multiple applications, such as autonomous driving, robot and crime prevention.

2.2.2 Traditional detector

Most of the early object detection algorithms were built based on manual made features with multiple complex model. Due to the lack of resources and image size, a number of speed up methods are required.

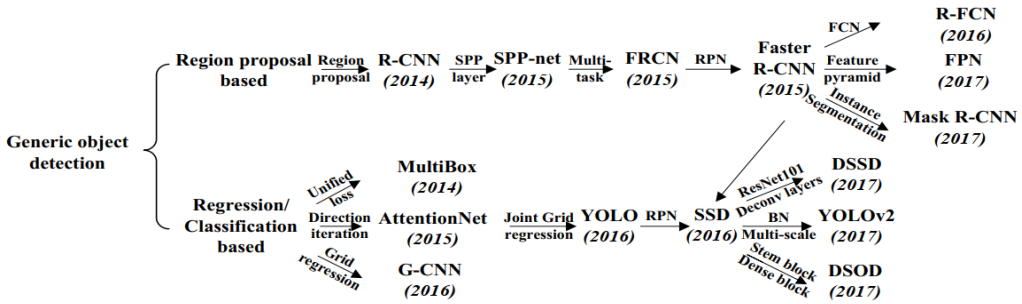


Figure 2.1: Object detection

In 2001, P. Viola and M. Jones achieved real-time face detection using sliding windows [3, 4]. The algorithm go through all possible locations and scales in an image to find the human face. It speeds up the computational process by involving three important techniques. The first one, integral image speeds up box filtering or convolution process using Haar wavelet as the feature representation of an image. The second one, feature selection using Adaboost algorithm to select a small set of features from a huge set of feature pools. The third one, A multi-stage detection paradigm reduces its computation by spending less time on background than on face possibility location. Although the algorithm is simple, it exceeds the current computational power of computer at that time.

Histogram of Oriented Gradients (HOG) was created in 2005 by N.Dalal and B.Triggs [5]. It is designed to be computed on a grid of equal cells and use overlapping local contrast normalization to improve accuracy. To detect object of different size, HOG resize the input image for multiple times to match with detection window size. It has been an important foundation of many object detectors and a large variety of computer vision applications.[6, 7]

2.2.3 CNN-based Detector

As the traditional methods showing their disadvantages by becoming more and more complex, the progress has been slow down, researchers tried to find an alternative to increase the accuracy and performance. In 2012, Krizhevsky et al. brought back the age of Convolutional Neural Network with a paper on classification with ImageNet [8]. As a DCNN can classify the image based on feature set, subsequent papers show their interest in the newly found method in object detection. Over the past decades, multiple network models has been proposed and studied to improve the accuracy in detection such as LeNet-5 [9], AlexNet [10], VGG-16 [11], Inception [12, 13], ResNet [14],etc. Studies also discover techniques that improve the training process and prevent overfitting, for example, dropout [10], Auto Encoder [15], Restricted Boltzmann Machine (RBM) [16], Data Augmentation [17] and Batch Normalization [18].

There are two main groups in CNN-based detection: ”two stages detecion” and ”one stage detection”. In the first group, firstly the image will be examined to generate proposals, and these proposals are delivered to another network for classification. In the second group, the object will be detected and classified directly within one network model.

2.2.3.1 CNN-based Two Stages Detection (Regional Proposal based)

Released in 2014 by Girshick et al.[19, 20], R-CNN is the first attempt in building a Convolutional Neural Network for object detection. The idea of R-CNN is divided into three main

stages:

- Proposals are generated by selective search.
- Proposals are resized to fixed resolution. These proposals are then used in CNN model to extract the feature map.
- Feature map is classified using SVMs for multiple classes to provide the final bounding box.

Despite having certain advantages comparing with traditional methods and bringing CNN back to practical use, R-CNN has some fatal disadvantages. The training has multiple stages and feature maps are stored separately thus increase time and space complexity. Moreover, The number of overlapped proposals are really high (over 2000 proposals for an image). The CNN model also requires a fixed size image, so any input must be resized and in certain occasions the object may get cropped, creating unwanted distortion.

Later in the same year, He et al. introduced a novel CNN architecture named SPP-Net [21] using Spatial Pyramid Matching (SPM) [22, 23]. The Convolutional Neuron Network is combined with a Spatial Pyramid Pooling (SPP) layer, make the network able to generate a fixed length feature representation without scaling the input image. The model removes the proposal overlapping and the need to resize image, however, it still require a multi-step training including feature extraction, network fine-tuning, SVM training and bounding box regressor. Additionally the convolution layers before the SPP cannot be modified with the algorithm shown in [21]

In 2015, Girshick proposed Fast R-CNN [24], a model with ability to do multi-task on classification and bounding box regression within the same network. Similar to SPP-Net, the whole image is processed with convolution layers to produce feature maps. Then, a fixed-length feature vector is extracted from each region proposal with a region of interest (RoI) pooling layer. Each feature vector is then fed into a sequence of fully connected layers before branching into two output, one is then used for classifier and the other encodes the bounding box location. Regardless of region proposal generation, the training of all network layers can be processed in a single stage, saving the extra cost on storage.

In the same year, Ren et al. introduced Faster R-CNN, a method to optimize Fast R-CNN further by altering the proposal generation using selective search by a similar network called the Region Proposal Network (RPN) [25]. It is a fully-convolutional network which has the ability to predict object bounds and scores at each position as the same time. With the proposal of Faster R-CNN, region proposal based CNN architectures for object detection can be trained in an end-to-end way. However, the alternate training algorithm is very time-consuming and RPN does not perform well when dealing with objects with extreme scales or shapes. As a result, multiple adjustments have been made. Some noticeable improvements are Region-based fully convolutional network (R-FCN) [26], Feature Pyramid Network (FPN) [27], and Mask R-CNN [28]. We will look in the details of these methods in the next chapter.

2.2.3.2 CNN-based One Stage Detection (Regression/Classification based)

Region proposal based frameworks are composed of several correlated stages, including region proposal generation, feature extraction, classification and bounding box regression. Even in Faster R-CNN or its variant, training the parameters is still required between the Region Proposal Network and detection network. As a result, achieving real time detection with Two Stages Detection is a big challenge. One Stage Detection, on the other hand, deal with the image directly by mapping image pixel to bounding box coordinates and class probabilities.

- You Only Look Once (YOLO): YOLO [29] was proposed by J.Redmon et al. in 2015 as the first entry to the One Stage Detection era. This network divides the image into regions and

predicts bounding boxes and probabilities for each region simultaneously. The YOLO consists of 24 convolution layers and 2 FC layers, of which some convolution layers construct ensembles of inception modules with 1×1 reduction layers followed by 3×3 conv layers. Furthermore, YOLO produces fewer false positives on background, which makes the cooperation with Fast R-CNN become possible. An improved version, YOLO v2,v3 and v4 was later proposed in, which adopts several impressive strategies, such as BN, anchor boxes, dimension cluster and multi-scale training.[30, 31, 32].

- Single Shot MultiBox Detector (SSD): SSD [33] was proposed by W. Liu et al. in 2016 as the second entry of the One Stage Detector. SSD introduces the multi-reference and multi resolution detection techniques which significantly improves the detection accuracy. The main difference between SSD and any other detectors is that SSD detects objects of many different scales on different layers of the network rather than detect at the final layer.
- RetinaNet: RetinaNet [34] uses a Feature Pyramid Network(FPN) with a CNN-based Backbone. FPN involves adding top level feature maps with the feature maps below them before making predictions. It usually involves upscaling the top level map, dimensionality matching of the map below using a 1x1 Convolution and performing element wise addition of both. RetinaNet achieves comparable result to Two Stages Detection while keeping much higher speed.
- Refinement Neural Network for Object Detection (RefineDet): RefineDet [35] is based on a feed forward Convolutional Network that is similar to SSD, produces a fixed number of bounding boxes and the scores indicating the presence of different classes of objects in those boxes, followed by the non-maximum suppression to produce the final result. RefineDet is formed by two inter-connected modules:
  - Anchor Refinement Module (ARM): Remove negative anchors and adjust the locations/sizes of anchors to provide initialization for the regressor.
  - Object Detection Module (ORM): do regressing object locations and predict multi-class labels based on the refined anchors.

  There are three core components in RefineDet: Transfer Connection Block (TCB) converts the features fro ARM to ODM; Two-step Cascaded Regression does regressing the locations and sizes of objects; Negative Anchor Filtering will reject well-classified negative anchors and reduce the imbalance problem. Further information will be discussed in the next chapter.

2.3 Diagram detection

In general, diagram detection can be grouped into two smaller areas: Online Diagram Recognition and Offline Diagram Recognition. In online recognition, the model is usually a RNN to recognize each stroke and generate candidate matches. Valois et al. [36] proposed a method for recognizing electrical diagrams. Each set of ink strokes is detected as a match with corresponding confidence factor using probabilistic normalization functions. Its disadvantage is the simplicity of the system and their low accuracy, prevent it from being used in real situations. Feng et al. [37] used a more modern technique in detecting electrical circuits. Symbol hypotheses generation and classification are generated using a Hidden Markov Model (HMM) and traced on 2D-DP. However, it has a drawback of complexity when the diagram and number of hypotheses is large, makes it impractical for real life cases. ChemInk [38], a system for detecting chemical formula sketches, categorizing strokes into elements and bonds between them. The final joint is performed using conditional random fields (CRF), which combines features from a three layers

hierarchy: inkpoints, segments and candidate symbols. Qi et al [39]. used a similar approach to recognize diagram structure with Bayesian CRF - ARD. These methods outperforms traditional technique, but the difficulty in joining the features using pairwise at the final recognition step make them harder for future adaptations. Coming to Flowchart recognition, after the release of the Online Handwritten Flowchart Dataset (OHFCD), multiple researches took place in resolving this dataset. Lemaitre et al [40]. proposed DMOS (Description and MOdification of the Segmentation) for online flowchart recognition. Wang et al. [41] used a max margin Markov Random Field to perform segmentation and recognition. In [42] they extend their work by adding a grammatical description that combines the labeled isolated strokes while ensuring global consistence of the recognition. Bresler et al. proposed a pipeline model where they separate strokes and text by using a text/non-text classifier then detect symbol candidates using a max-sum model by a group of temporally and spatially close stroke. The author also propose an offline extension that uses a preprocessing model to reconstruct the strokes from flowchart [43, 44].

While online flowchart recognition detects candidates based on stroke, offline flowchart recognition receives an image from the user and use it to generate the diagram. It might be possible to reconstruct online stroke from offline data [45], however, that preprocessing step might not be necessary because we can recognize the whole diagram structure independently with strokes. As online recognition seem to attract more researchers, there has not been many studies about offline detection. A. Bhattacharya et al. [46] uses morphological and binary mapping to detect electrical circuit. Although it may work on a smaller scale, using binary mapping may not be able to detect curve or zig-zag lines. Julca-Aguilar and Hirata proposed a method using Faster R-CNN to detect candidates and evaluate its accuracy on OHFCD. The model is able to detect components in the diagram, including arrows, however, it is not able to detect the arrow head.

2.4 Handwriting character recognition techniques for English alphabets

Handwriting recognition is a study domain in fields of image processing and pattern recognition which is popular and complicated in the current day. It contributes immensely to the sequence of a computerization technique and could enhance the interface among human and machine in so many purposes. This process primarily start with preprocessing procedure which is followed by line segmentation, normalization, feature extraction, recognition and transcriptions according to [47]. The preprocessing includes the process that improves the structure of the image which is more suitable in segmentation. In the segmentation procedure, the raw image is segmented into single characters and after that, the procedure reconstruct each character in m x n pixels to the training network. Next is about feature extraction technique, which affects recognition percentage. There are numerous of feature extraction methods which are mentioned in the research of [48]. Additionally, according to [47], the frequently used feature extraction techniques are Contour profiles, Deformable templates, Fourier descriptors, Gabor features, Gradient feature, Graph description, Geometric moment invariants, Template matching, Unitary Image transforms, Projection Histograms, Zoning, Zernike Moments and Spline curve approximation.

2.4.1 Preprocessing

In preprocessing procedure, there are many actions needs to be executed, which are noise removing, binarization, edge detection, dilation and filling, processed image for feature extraction according to [47]. In noise removing phase, the reason for this phase is that scanned documents can be contaminated with dust, dots, or lines, which are classified as noise that has a negative impact on the recognition results. Therefore, image is applied with smoothing and non linear operations to improve the image quality. In addition, for text line which is not horizontally align during scanning process, the skew correction can be applied on the document or line level. In the preprocessing procedure, skew correction is applied on the document level.

2.4.2 Line segmentation

In line segmentation, this phase separates the image whose structure in a sequence of alphabet characters into sub image which contains a single character by conveying a number to every alphabet utilizing a labeling procedure. This labelling provides knowledge about number of characters in the image. Every single character is restructured into 90*60 pixels for categorization and recognition stage.

2.4.3 Normalization

After line segmentation, normalization is performed to eliminate the differences that complicate the categorization and decrease the recognition percentage of the similar character or word across different writers. The most general basis of variability in handwritten alphabet images is incline and text volume according to [47]. Slant correction techniques is applied to correct the inclination in the writing style. By applying a shear transformation, the writings slant is transformed into its upright position. A shear transformation is applied in many directions. For each direction, the transformed image data are added with pixel values of the same image that are vertically shifted by distance d and –d.

2.4.4 Feature extraction

In feature extraction step, there are two main approaches which are used holistic and analytic [47]. In holistic recognition, each word is considered to a class and is recognized as whole word. On the other hand, the other approach is based on character segmentation-free recognition. Based on [47], features used in holistic offline recognition systems are categorized into high, middle, and low levels. The high level removes the features from the entire phrase image, the middle level removes features from the letters, and the low level removes features from the sub letters. There are various types of holistic features which are represents in [47]. In analytic recognition schemes, the image is representing as a chain of characteristics. This scheme uses a slicing window which is shifted from left to right. At each position n, a characteristic vector fn is calculated from just the pixel within the slicing window. Two major slicing window scheme characteristics are profile characteristics and window characteristics [47].

2.4.5 Recognition

The English alphabets might be scripted in various size, direction, width, arrangement and measurement [47]. Thus, recognition of English alphabets is extremely difficult problem. There

are some frequently used learning models in this recognition process such as HMM (Hidden Markov Model), NN (Neural Network) and so on.

Chapter 3

Background

3.1 Preprocessing

Pre-processing plays an important role in image detection. The aim of pre-processing is improving the construct the raw image and enhancing features of the image which is appropriate in further processing.

3.1.1 Grayscale image

For flowchart, color is not a important feature to keep in recognition procedure and color also causes the detection algorithm more complex. Therefore, we need to remove this feature by converting the image into grayscale image. In this part, we represent some fundamental method to convert a RGB image into grayscale image.

- Average method is the most simple one, we only need to take the average of three colors red, green and blue $((R+G+B)/3)$
- Lightness method averages the most prominent and least prominent colors $((max(R,G,B)+min(R,G,B))/2)$
- Luminosity method forms a weighted average to account for human perception. Human are more sensitive to green than other colors, so green is weighted most heavily $(0.299R+0.587G+0.114B)$

3.1.2 Histogram equalization

Histogram equalization is a technique used to improve contrast in image by spreading out the most frequent intensity values. By accomplishing this, histogram equalization allows the image's areas with lower contrast to gain a higher contrast. This technique includes steps as below:

- Calculate the histogram $H(i)$ of the image by counting the total number of pixels which has similar intensity value.
- Calculate the cumulative distribution $H'(i)$ from $H(i)$ with the formula: $H'(i) = \sum_{j=0}^{i} H(j)$
- Normalize $H'(i)$ such that the maximum value equals the maximum value for the intensity of the image which is 255
- Mapping lightness values with $H'(i)$ to obtain the equalized image with the formula: $equalizedImage(x,y) = H'(srcImage(x,y))$

3.1.3 Noise filtering

As we introduce about noise in the Chapter 1, noise filtering techniques plays an important role in the pre-processing procedure. In this part, we give some fundamental approaches such as Gaussian filter which is simple and suitable for Gaussian noise and Median filter which is better than Gaussian filter in preserving edges according to [49]. In the Chapter 6, we are going to compare these two approaches with a captured image in our project.

3.1.4 Binarization

Image binarization is the process of converting a grayscale image to a black-and-white image, essentially reducing the information contained within the image from 256 shades of gray to 2 which are only black and white. The approach used in this part is Adaptive Thresholding. In simple thresholding, the threshold value is global for all the pixel in image, while Adaptive thresholding is the method where the threshold value is calculated for smaller regions and therefore, there will be different threshold values for different regions. Therefore, this approach gives a better result than simple thresholding.

3.2 Flowchart detection

We decided that there would be two algorithms in our project. One detection will be faster and one will be more accurate. For the Two Stages Detection, we are planning to use Faster R-CNN with RPN and RoI Align. For the One Stage Detection, we are trying to use both YOLO v4 and RefineDet and select whichever yield better result.

3.2.1 Faster R-CNN

As introduced in earlier section, Faster R-CNN extends Fast R-CNN by introducing Region Proposal Network A Faster R-CNN consists of three parameters: Backbone CNN, RPN and Classification model.

3.2.1.1 Backbone CNN

The Convolutional Neural Network is an important part in the algorithm. The better the CNN is, the higher accuracy the model will have. Currently, we are making an experiment on three different network: ResNet50, Inception v4 and Inception ResNet v2. The reason we decide to do so is because ResNet50 is a lightweight model and Inception v4/Inception ResNet v2 provides higher results with computational tradeoff according to this page.

3.2.1.2 Regional Proposal Network

RPN is a small network that slides over a convolutional feature map which is the output by the last convolution layer. It has a classifier and a regressor. In the first step, the input image goes through a CNN to give the output as a set of feature maps on the last layer. Next, a sliding window is run spatially on these feature maps. For each sliding window, a set of 9 anchors are generated which all have the same center but with 3 different aspect ratios and 3 different scales. Furthermore, for each of these anchors, a value p* is computed which indicated how much these

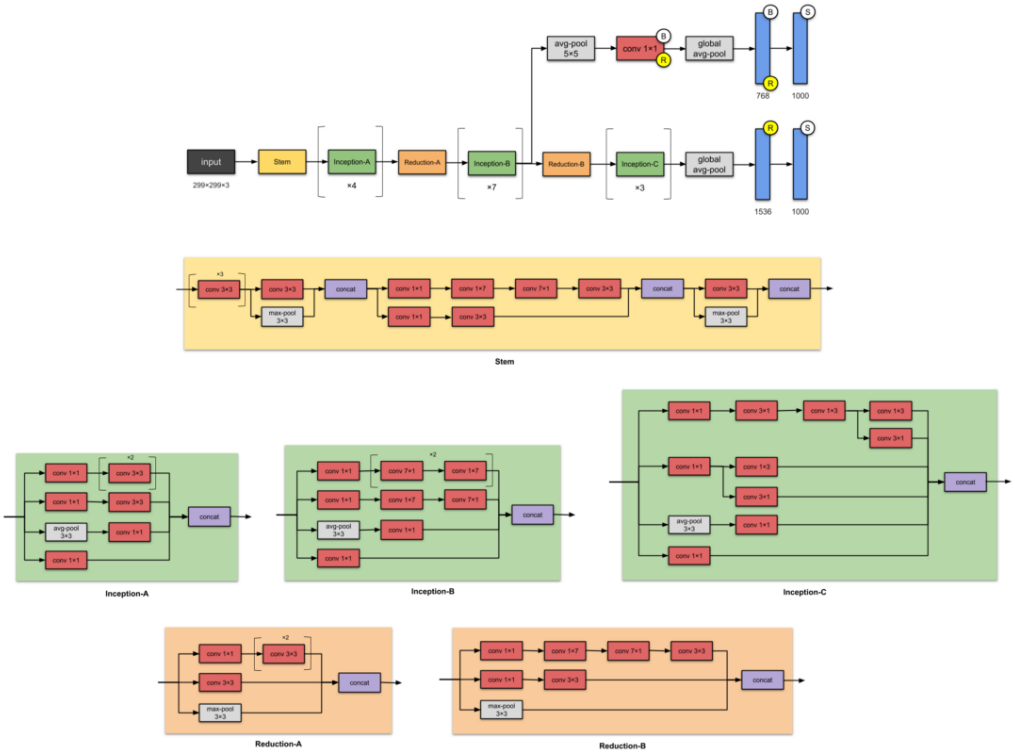


Figure 3.1: Inception v4 model

anchors overlap with the ground-truth bounding boxes. p* is calculated by:

$$p^* = \begin{cases} 1 & \text{if } k \geq 0.7 \\ -1, & \text{if } k \leq 0.3 \\ 0 & \text{otherwise} \end{cases} \text{where} k = \frac{\text{Anchor} \cap \text{GroundTruthBox}}{\text{Anchor} \cup \text{GroundTruthBox}}$$

Lastly, The 3×3 spatial features extracted are fed to classification and regression. The output of regressor is the predicted bounding box, the output of classifier is the probability that the box contain an object or background.

$$\begin{aligned} t &= [(x - x_a)/w_a, (y - y_a)/h_a, log(w/w_a), log(h/h_a)] \\ t^* &= [(x^* - x_a)/w_a, (y^* - y_a)/h_a, log(w^*/w_a), log(h^*/h_a)] \\ \text{Loss function:} L(p_i, t_i) &= (1/N_{cls}) L_{cls}(p_i, p_i^*) + \lambda (1/N_{reg}) L_{reg}(t_i, t_i^*) \end{aligned} \tag{3.1}$$

p* with regression term in the loss function ensures that if and only if object is identified as yes, then only regression will count, otherwise p* will be zero, so the regression term will become zero in the loss function.

3.2.2 Feature Pyramid Network

Feature pyramids is a component in recognition systems for detecting objects at different scales. It consists of two main components:

- Bottom-up Pathway: The bottom-up pathway is the feed-forward computation of the backbone CNN. It is defined that one pyramid level is for each stage. The output of the last layer of each stage will be used as the reference set of feature maps for enriching the top-down

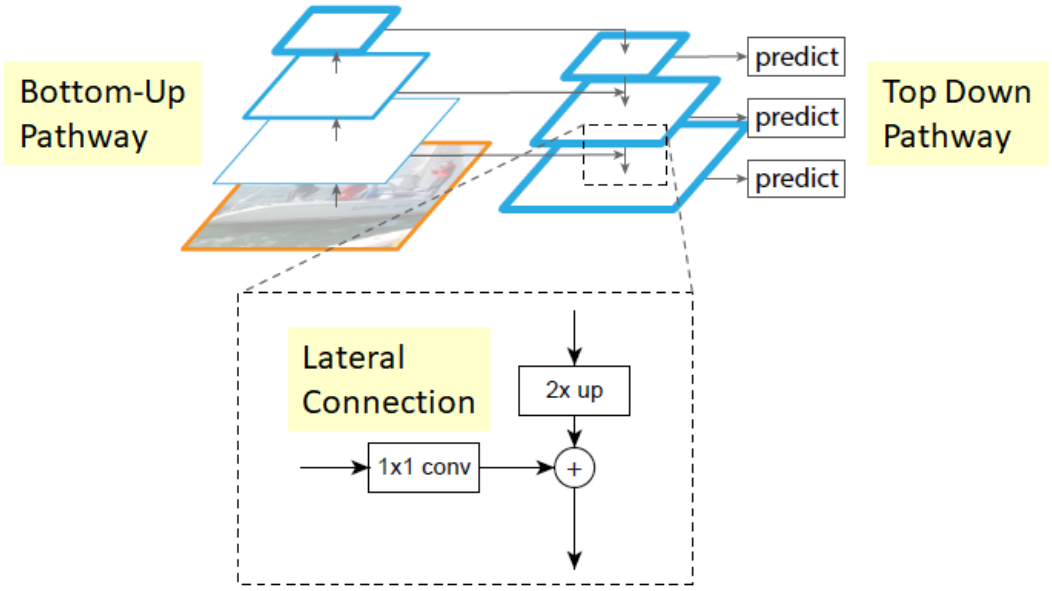


Figure 3.2: Feature Pyramid Network

pathway by lateral connection. Normally, one layer has its size equal to 1/2 of the previous layer.

- Top-down Pathway and Lateral Connection: The feature map at the highest layer in the bottom-up pathway is brought to the corresponding one of the Top-down pathway. For every layer under it, a lateral connection is constructed by the following steps: The upper layer is up by a factor of 2. The corresponding feature map from bottom-up is undergone through a 1x1 convolution to reduce dimension. Finally, both feature maps from bottom up and top down are merged by element-wise addition. After the final layer is computed, or the algorithm stop when reaching a certain level, all merged layers go through a 3x3 Convolution layer to generate the final feature map.

3.2.3 Region of Interest Align

Mask R-CNN introduces a new algorithm called Region of Interest Align. It replaces the RoI Pooling in Faster R-CNN. RoI Align deals with the round-off errors that RoI Pooling cannot. The difference was to introduce bi-linear interpolation when calculate the pixel's value for the floating point coordinate. For example (69.5, 420.7). This is a floating point coordinate. We however know the what pixel value for (69, 420) and (70, 421). So to estimate (69.5, 420.7) we can use a technique used in many image processing tricks that is called bi-linear interpolation. The step are processed as follow:

- The RoI pooling layer divide the feature map into M by N grid. For each small grid, the unit is then sampled K times. For the Mask R-CNN paper they used K=4 for best result.
- Divide each unit equally by 4 means finding the center pixel values for the these 4 regions in the unit. Of course these centers are floating point based. Therefore we use bi-linear interpolation to predict its value.
- After bi-linear interpolation, we perform max pooling on these 4 samples to output the unit's value.

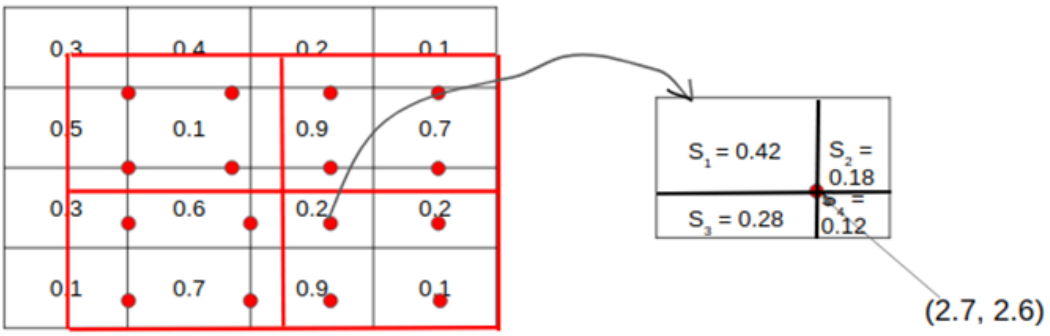


Figure 3.3: Region of Interest Align

3.2.4 YOLO

3.2.4.1 YOLO v1

As figure 3.4 YOLO divides the input image into an S $\times$ S grid. For each object that is present on the image, one grid cell is said to be "responsible" for predicting it. That is the cell where the center of the object falls into.

Each grid cell predicts B bounding boxes as well as C class probabilities. The bounding box prediction has 5 components: (x, y, w, h, confidence). The (x, y) coordinates represent the center of the box, relative to the grid cell location. These coordinates are normalized to fall between 0 and 1. The (w, h) box dimensions are also normalized to [0, 1], relative to the image size. Confidence scores are calculated as Pr(Object) multiply with IOU_{pred}^{truth}.

At the same time, regardless of the number of boxes, C conditional class probabilities (Pr(Class(i)|Object)) should also be predicted in each grid cell. It should be noticed that only the contribution from the grid cell containing an object is calculated. At test time, class-specific confidence scores for each box are achieved by multiplying the individual box confidence pre-

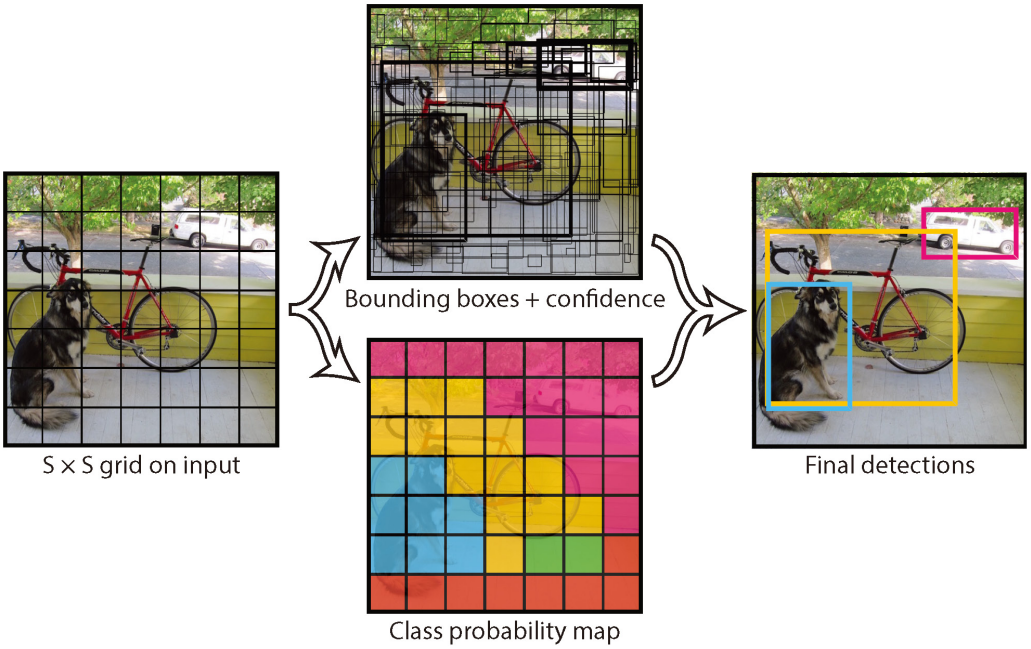


Figure 3.4: YOLO model

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2$$

$$+\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

Figure 3.5: Lost function of YOLO v1

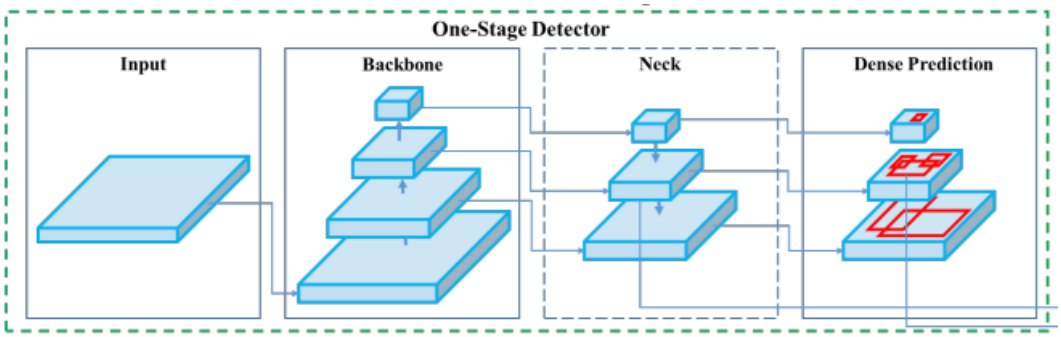


Figure 3.6: YOLO v4

dictions with the conditional class probabilities as:

$$Pr(Object) * IOU_{pred}^{truth} * Pr(Class_i|Object) = Pr(Class_i) * IOU_{pred}^{truth}$$

During training, the following loss function is considered as figure **??**

3.2.4.2 YOLO v4

YOLO v4 is the latest entry in this type of detector. YOLO v4 achieves really high accuracy with high speed. To get these results, YOLO v4 combines multiple features such as Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT) and Mish-activation, Mosaic data augmentation, DropBlock regularization, and CIoU loss. YOLO v4 consists of three main stages. The Backbone is a CNN model like ResNet, VGG are trained and tuned on detection dataset. THe

neck is many extra layers that go in between the backbone and head. They are used to extract different feature maps of different stages of the backbone. The neck layer can be either FPN or Bi-FPN, which will be discussed on RefineDet. The head is a network doing the detection (classification and regression) of bounding boxes.

3.2.5 RefineDet

RefineDet is another candidate for us to use or improve in this project. From the previous chapter, we know that RefineDet is formed by two inter-connected modules: Anchor Refinement Module (ARM) and Object Detection Module (ORM). The system has three main element:

- Transfer Connection Block is responsible for converting features of different layers from the ARM into the form required by the ODM, making ODM able to share features from ARM. Another function of the TCBs is to integrate large-scale context by adding the high-level features to the transferred features to improve detection accuracy.
- Two-Step Cascaded Regression: Most current method relies on one-step regression based on multiple feature layers. RefineDet is not the same, it uses a Two-Step Cascaded Regression strategy to regress the locations and sizes of objects.
- Negative Anchor Filtering: In training phase, for a refined anchor box, if its negative confidence is larger than a preset threshold θ (i.e., set θ = 0:99 empirically), that box will be discarded. This helps refined hard negative anchor boxes and refined positive anchor boxes to train the ODM. In the inference phase, if a refined anchor box is assigned with a negative confidence larger than θ, it will be discarded in the ODM for detection.

Chapter 4

Proposed system

4.1 Project scope and restrictions

- The topic focuses on building mobile apps on Android platform using Flutter and the main programming language is Dart and doesn't support other mobile platforms like iOS, Windows Phone.
- User can only convert hand drawn diagram to digital version if internet connection is available.
- The web server can serve 100 clients at the same time.
- The system only supports converting and editing of small charts (in an A4 page, medium size text) and does not support importing files from other platforms.
- The system only create and editing flowchart.
- The system only supports storing chart files in .json format and exporting to images (png, jpg), text (pdf) and .drawio files.
- The flowchart contains maximum 20 symbols not including arrows
- The system do not allow to take picture with the wrong direction.
- The flowchart must not have three or more arrows intersect at one point. It also must not have two arrows having the same edge.
- The flowchart must be drawn with ball pen on a fresh white paper.
- Arrow in flowchart must have an arrowhead.
- There must not be isolated text/symbol. Furthermore, strikethrough text is restricted.

4.2 Application features

The application runs on mobile devices so that users can easily convert a graph on the fly as long as they are connected to the internet. It also requires storage access and camera access permission in order to use the scanning function and temporarily saving it in the phone.

4.2.1 Sign up and Login

When users open the app for the first time, they will be offered to create an account to continue. To simplify the sign-up process, it only requires the user's email and password. Other information can be updated later on. In addition, users can also use their Google account to sign in without creating their own account for this app. In case users had already forgotten the

password, they can retrieve it by providing their account email and the system will send an email to verify and include further information for changing the password.

4.2.2 Create diagram

After successfully login in, users can create a new diagram using one of these methods:

- Directly Scanning: The application activates the device's camera so user can point at the drawing and take a picture of it.
- Import from image: User selects an image from device's gallery.
- Create blank: User can create a blank diagram and add elements then.

4.2.2.1 Convert to digital

If user chooses the first two options then after taking/choosing the picture he/she can re-take or re-adjust the picture (rotate, flip, or crop) before finishing this step. After that, the result will be displayed and provide some extra options: save, modify, and delete. Save option will require inputting the diagram name and choose the saving location. The diagram can be saved online on the web server or offline on the phone. Each user can only save a limited number of diagrams online, if a user is running out of saving slot then the only option is to store them locally. The main difference between saving online and offline is only the diagrams on the web server can be shared and have a detailed change history. Modify option is used if user is not satisfied with the result of the auto-detection algorithm and want to change it. This option will send them to the modify page where they can edit it before saving. Note that if user leaves this page without saving it, the diagram will be lost. Delete option shows a confirmation message and offers a retry.

4.2.2.2 Create from blank

If "Create blank" is chosen then a name for new diagram is required, then user will be sent to modify page where already has a default diagram created.

4.2.3 Adjust diagram

All the created diagrams will be displayed on the main screen of the app, user can select one of them to modify, delete, rename, view history change, export, and share. Only diagrams that are stored online have the share option and view history change option.

On the modify page, user can change elements' position, create new ones, or delete them. All elements supported will be listed later. All the attribute that user can change within an element is size, title, color, and type of element. All of the changes will be stored so that the user is able to undo or redo the action. Auto-saving is not supported, user needs to save manually. If user exits the app without saving, all of the progress will be lost. "Change history" function requires to store a version of the old diagram every time a new one is saved into the web server. If the diagram is shared within a group, all of the information about the user who makes changes is also stored.

4.2.4 Share and set permission

Users can also share the diagram with others. However, only the owner can share their product. All of the shared diagrams will be in another tab on the home screen. To share a

diagram, user needs to input the receiver's email and set their permission to "Read-only" or "Can edit". The Owner can also revoke the share permission by deleting the other user from the share list.

4.3 Requirements

4.3.1 Functional requirement

- System is able to detect the flow-chart diagram from a hand-drawing draft.
  - System is able to detect the node type and its title.
  - System is able to detect the relationship between the diagram's symbols and check for incorrect relationship.
  - System is able to detect the relationship between the diagram's symbols and check for incorrect relationship.
  - English is the main supported language.
- System is able to re-create the diagram into digital version to display in the mobile application, and allow user to modify.
- The input can be directly taken from the camera of the device or uploaded from the storage.
- (Optional) System has the ability to convert the diagram to some popular diagram file format so that user can import it from tools like drawio, lucidchart, ...

4.3.2 Nonfunctional requirement

- System is written with Dart (with framework Flutter), C++.
- System is able to detect diagram within 8 seconds.

4.3.3 Hardware requirement

- Mobile application require device must be installed with android OS version 7.0 or above
- Recommend requirement for web server:

Chapter 5

System design

5.1 System Architecture

At first, this application has two system options: One that requires no additional server to run, all of the service is installed locally on the user's devices. Since our core service, the diagram detection algorithm is written with C++ then it can run on any system. Then we intended to build an application system that can handle the whole process of pre-processing image and create the result without the internet connection. However, as our target is to support all the device that runs android 7.0 and above (that means all smartphones manufactured from 2016 to present and some of them can be older), it could lead to a performance problem because many of them were outdated 2 or 3 years. Another issue is sharing between users will have to be dependent on a third-party service. Which is not convenient and may lower the user experience.

The second one is to build a REpresentational State Transfer Application Programming

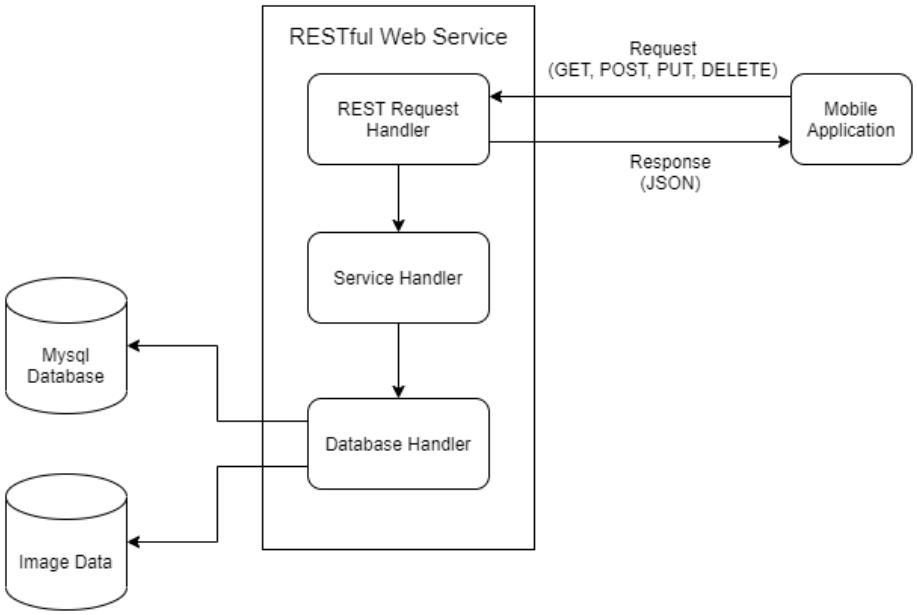


Figure 5.1: System Architecture Design

Interface (REST API) service, run on a node.js server, which will handle all the requests sent by users and store all the data on a system database so that users can now store all of their data online without worry about their phone internal storage. the web server can be divided into three main components: Request handler, Service handler and Database handler. All of the detection and management will be mainly process in the second component, then all of the data will be transfer to a My SQL database, including user data and diagram file. However, all of the uploaded images that sent to the server to process will be then stored in the image storage separated from main database for additional data to improve detection algorithm in the future. The downside of this approach is now in order to use the app, the application has to be stably connected to the internet to make any changes to their data (create, edit, remove,...). Another challenge for us is to optimize the server so it can process a lot of access at the same time.

5.2 Framework

5.2.1 Flutter

Flutter is a user interface toolkit developed by Google for building beautiful, natively compiled applications for mobile, web, and desktop from a single code base [50]. Dart is the main programming language used in Flutter.

One of the most beloved of Flutter that finally made it the framework we use in this thesis is it can create great and modernized user interfaces that really suit our design code for this project: simplistic but easy to use. Although the result can be used to build both android and iOS (potentially web) applications, the target of this project is to focus on one platform at first and continue to grow in the future. Despite being a recently released framework built for mobile development, Flutter provides a lot of features that speed up the development process as well as a growing community to share and learn.

However, there is also a downside of it, since it was born to build an application that can run on many different platforms, the performance loss is inevitable compared to natively built applications. Even though Flutter has been optimized and guarantee 60 frames per second but it still heavily depends on developers to make good use of available performance resources. Another one is the application file built by Flutter tends to have a bigger file size and also consume a lot more storage to install. A normal solution for this is to reduce image resolution and use less graphic and animations in the app, but Flutter still shows a poorer result than the native app.

5.2.2 Nodejs

Nowadays, Node.js is one of the most popular JavaScript Run time Environment as it is an open-source, cross-platform, and powerful tool that is used for many server-side projects [51]. One of the biggest advantages is using JavaScript, which supports asynchronous programming. In the Node.js server, all requests will be handle by a single process without creating any thread. When a process requires queries in the database, the system allows pausing it until the queries are finished, in the meantime, other jobs can be handled without wasting a whole thread. As the number of clients increases over time, this can be a great feature to scale up the system.

5.3 Database Design

For each user, besides user name, email and password are essential information, an account must also have an account type to indicate whether it is a system account or from another platform account (such as Google). If so, it required another field to store the account ID key. If the account is created in the application then the password needs to be hashed and salted, which required 2 fields for salt string and encoded password. In order to perform any action with the server from the client-side, it requires a key for authentication each time the device calls an API, which will be changed each time a user logins. This key will be expired after half an hour of inactivity. Each time an API is called, the expired time will be refreshed. The expired time is stored with the user account for comparison each time this key is used. With this design, the system cannot keep track of the number of times a user logged in or how many times the API was called with a specific user key. However, all of the activities on the server will be recorded in other tables (creating and uploading diagram, send review), it will be unnecessary to create a table for login records.

Each diagram has to belong to a user, diagrams can be shared between users with different permission like ”Owner”, ”Edit-Only”, and ”Read-only”.

There can be two types of diagrams, the ”converted” diagram and the ”created from blank” one. For the first type, it requires a field to store the image path in the image storage.

As each diagram may have many versions, each time the user uploads a new one, it will be stored with a new ID and also have an upload type to distinguish between new upload and revert (as the user is able to view the change history, they may also revert back to an older version), then it will need an additional field to store the reverted version in order not to store many same version of the same diagram file.

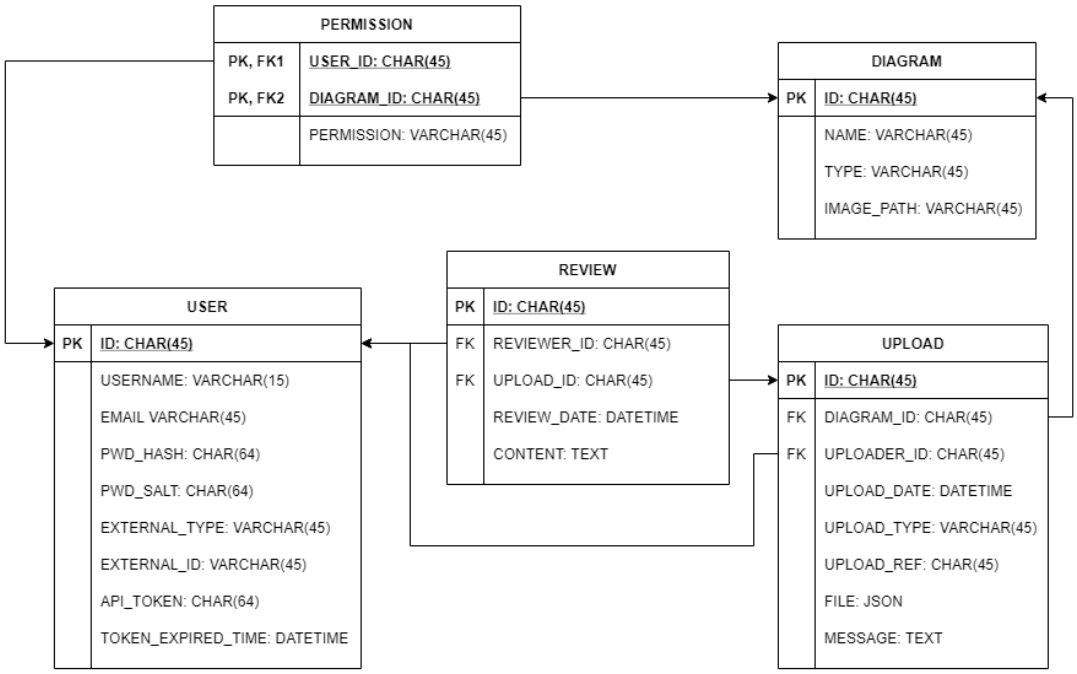


Figure 5.2: Database Design

5.4 Diagram File Design

After having consulted some types of diagrams, we decided to go with .json file since it is widely used and supported by many platforms. It is also able to be stored directly in the My SQL database using JSON data type. Although the size still depends on the server configuration, this is one of the most convenient file types to transfer between the web server and the client devices. The file will have 4 main components: ID, name, an array of symbols, and an array of arrows.

Figure 5.3 shows the json file design used for construct the diagram used in the application.

5.5 Use case Design

5.5.1 Use case diagram

Figure 5.4 shows the system use case design for the application.

```
{
    "id": ID of diagram,
    "name": Name of diagram,
    "symbol":  Array of symbols
    [{
        "id": symbol ID,
        "dx": percentage of width,
        "dy": percentage of height,
        "content": symbol text,
        "backgroundcolor": HEX value,
        "bordercolor": HEX value,
        "contentcolor": HEX value,
        "type": Symbol type
    }],

    "arrow": Array of arrows
    [{
        "id": arrow ID,
        "start": start symbol ID,
        "end": end symbol ID,
        "color": HEX value,
        "content": arrow text,
        "contentcolor": HEX value,
        "type": arrow type
    }]
}
```

Figure 5.3: JSON file design

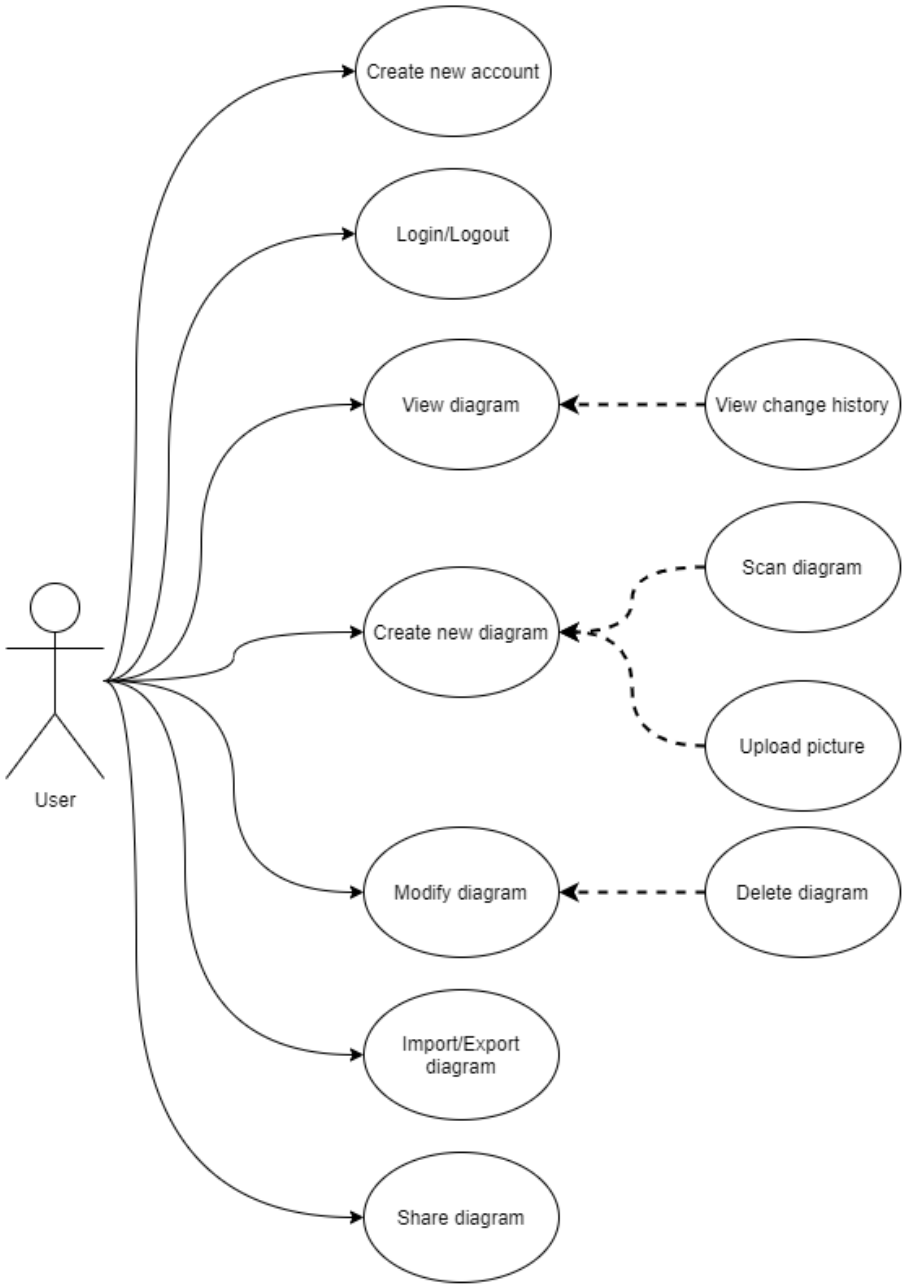


Figure 5.4: Use case Design

5.5.2 Use case detail

Use case name	Table reference
Login.	Table 5.2
Sign up.	Table 5.3
Create new diagram.	Table 5.4
Scan diagram with camera.	Table 5.5
Scan from image.	Table 5.6
Import file.	Table 5.7
Export file.	Table 5.8
Modify diagram.	Table 5.9
Delete diagram.	Table 5.10
Logout.	Table 5.11

Table 5.1: Use case List

Use case ID:	UC-1
Use Case Name:	Login.
Triggering Event:	The user open the app and in login screen.
Brief Description:	System verify user's inputted username and password and go to home screen.
Actors:	The user
Preconditions:	• Application has internet connection. • The user has a system account. • The user has not logged in with Google account.
Post-conditions:	• The user is logged in to the system. • The user is able to perform action that requires the web server.
Normal flows:	1. The user opens the app. 2. System show login form including username and password. 3. The user fill in the form and login. 4. System authenticates user. 5. The user is logged in and has access to the system.
Alternative flows:	• 2a The user has already logged in with Google account – 2a1 System automatically authenticates user in with Google account. – 2a3 Use case continue with step 5. • 3a The user chooses to log in with Google account – 3a1 System shows Google logging page. – 3a2 The user enters essential information. – 3a3 System receive Google credential. – 3a4 Use case continue with step 4. • 4a The user input an invalid username or password. – 4a1 System shows an error message. – 4a2 The user re-input information. – 4a3 Use case continue with step 4.
Exception:	• 2a The user logged in with Google account but the credential is incorrect. – 2a1 System shows an error message. – 2a2 Use case continue with step 1.

Table 5.2: Use case: Login.

Use case ID:	UC-2
Use Case Name:	Sign up.
Triggering Event:	The user choose ”Sign up” in login screen.
Brief Description:	The user want to create a new system account.
Actors:	The user
Preconditions:	• The user has an email account. • Application has internet connection. • The user has not logged in the system.
Post-conditions:	• A new account is registered in the system with distinct user name. • The user is logged in and has access to system function.
Normal flows:	1. The user open the app and choose ”Sign up” function. 2. System show the register form including username, email, password. 3. The user enters the essential information. 4. System check if all of the information are valid and create a new account. 5. System authenticates user. 6. The user is logged in and has access to the system.
Alternative flows:	• 4a The user input an invalid username or password. – 4a1 System shows an error message. – 4a2 The user re-input information. – 4a3 Use case continue with step 4. • 4a The user input an already existed username. – 4a1 System shows an error message and provide login option. – 4a2 The user choose login. – 4a3 Use case continue with table 5.2.
Exception:	N/A

Table 5.3: Use case: Sign up.

Use case ID:	UC-3
Use Case Name:	Create new diagram.
Triggering Event:	The user wants to create a new diagram.
Brief Description:	At home screen, user wants to create a new diagram.
Actors:	The user
Preconditions:	
Post-conditions:	• A diagram is detected. • A new temporary file is created.
Normal flows:	1. The user opens the app. 2. The user chooses to create a blank diagram. 3. The user input diagram name. 4. System show file browser with default saving location. 5. The user chooses location to save diagram. 6. System create new file in storage and go to modify page.
Alternative flows:	• 2a The user choose to scan a new diagram directly. – 2a1 Use case continue with table 5.5. – 2a2 Use case continue with step 3. • 2b The user choose to scan a new diagram from image. – 2a1 Use case continue with table 5.6. – 2a2 Use case continue with step 3.
Exception:	• 3a The user do not input diagram name. – 3a1 System uses default name (New Diagram). – 3a2 Use case continue with step 4. • 3b The user delete default name and leave name blank. – 3b1 System shows an error alert and let user try again. – 3b2 Use case continue with step 3. • 5a if the app has not been granted access to storage. – 5a1 System shows access permission notification. – 5a2 The user allow storage permission. – 5a3 Use case continue with step 6. • 5b If system cannot create a new file. – 5b1 System show error and return to home screen. – 5b2 Use case stop

Table 5.4: Use case: Create new diagram.

Use case ID:	UC-3.1
Use Case Name:	Scan diagram with camera.
Triggering Event:	The user wants to scan a new diagram with camera.
Brief Description:	The user chooses to scan device camera in create options.
Actors:	The user
Preconditions:	
Post-conditions:	• A diagram is detected.
Normal flows:	1. The user takes a picture of the diagram. 2. The user adjusts the boundary of the diagram. 3. System shows result. 4. The user chooses “Done”. 5. System continue with previous use case.
Alternative flows:	• 2a The user want to retake the picture. – Use case continue with step 1.
Exception:	• 3a If there is no diagram detected. – 3a1 System show an error. – 3a2 The user choose to retake the picture. – 3a3 Use case continue with step 1.

Table 5.5: Use case: Scan diagram with camera.

Use case ID:	UC-3.2
Use Case Name:	Scan from image.
Triggering Event:	The user clicks on “Scan from image” button in create options.
Brief Description:	The user chooses to scan a new diagram from an image.
Actors:	The user
Preconditions:	N/A
Post-conditions:	• A diagram is detected.
Normal flows:	1. System shows the storage browser. 2. The user chooses a picture. 3. System shows the full picture. 4. The user adjusts the boundary of the diagram. 5. The user chooses “Done”. 6. System save the result and continue with previous use case.
Alternative flows:	N/A
Exception:	• 2a The user chooses a file that is not a picture. – 2a1 System show a warning – 2a2 The user choose to cancel. – 2a3 Use case stop. • 3a there is no diagram detected. – 3a1 System show an error. – 3a2 The user choose to retake the picture. – 3a3 Use case continue with step 1.

Table 5.6: Use case: Scan from image.

Use case ID:	UC-4
Use Case Name:	Import file.
Triggering Event:	The user clicks on “Import file” button.
Brief Description:	The user is in home page and want to import a file in storage to the app.
Actors:	The user
Preconditions:	N/A
Post-conditions:	• A new diagram is imported and show in home screen.
Normal flows:	1. The user chooses to import from local storage. 2. System shows the storage browser. 3. The user chooses a file. 4. System save file location and show new diagram in home screen.
Alternative flows:	• 2a The user choose to import from Google drive. – 2a1 The user login to Google Drive. – 2a2 System shows Google Drive file browser. – 2a3 Use case continue with step 3.
Exception:	• 3a The user chooses a file that is compatible or not accessible. – 3a1 System show a warning. – 3a2 The user choose to cancel. – 3a3 Use case stop.

Table 5.7: Use case: Import file.

Use case ID:	UC-5
Use Case Name:	Export file.
Triggering Event:	The user clicks on “Export” button in diagram options.
Brief Description:	The user wants to export a diagram.
Actors:	The user
Preconditions:	N/A
Post-conditions:	• A new diagram file is created.
Normal flows:	1. The user selects a diagram and choose “Export”. 2. The user selects “Export as an image”. 3. The user chooses save location and select “Done”. 4. System create an image of selected diagram.
Alternative flows:	• 2a The user select “Export as .pdf file”. – 2a1 The user chooses save location and select “Done”. – 2a2 System create an pdf file of selected diagram. – 2a3 Use case stop. • 2b The user select “Export as .drawio file”. – 2b1 The user chooses save location and select “Done”. – 2b2 System create a drawio file of selected diagram. – 2b3 Use case stop.
Exception:	• 3a If the diagram is blank, show a warning and allow user to cancel. – 3a1 The user cancel the operation. – Use case stop. • 3b if the app has not been granted access to storage. – 3b1 System shows access permission notification. – 3b2 The user allow storage permission. – 3b3 Use case continue with step 4.

Table 5.8: Use case: Export file.

Use case ID:	UC-6
Use Case Name:	Modify diagram.
Triggering Event:	The user clicks on “Modify” button in diagram options.
Brief Description:	The user wants to modify a diagram showed in home screen.
Actors:	The user
Preconditions:	N/A
Post-conditions:	• All changes is recorded and uploaded to server.
Normal flows:	1. The user selects a diagram. 2. System shows diagram options. 3. The user selects “Modify”. 4. System go to modify page. 5. The user performs changes to the diagram. 6. The user selects “Save”. 7. System saves all changes and go back to home screen
Alternative flows:	• 6a The user select ”Discard”. – 6a1 Systems show confirm message. – 6a2 The user select ”Yes”. – 6a3 Systems delete all changes and return to home screen. – 6a4 Use case stop.
Exception:	• 7a System cannot save file. – 7a1 System show saving error. – 7a2 The user selects cancel. – 7a3 Systems delete all changes and return to home screen. – 7a4 Use case stop. • 7b if the app has not been granted access to storage. – 7b1 System shows access permission notification. – 7b2 The user allow storage permission. – 7b3 Use case continue with step 4.

Table 5.9: Use case: Modify diagram.

Use case ID:	UC-7
Use Case Name:	Delete Diagram.
Triggering Event:	The user choose ”Delete” in diagram options.
Brief Description:	The user want to delete a diagram from home screen.
Actors:	The user
Preconditions:	• The user is in home screen. • The user is the owner of the diagram. • The diagram is no longer shared with any other user.
Post-conditions:	• The diagram is deleted in the database.
Normal flows:	1. The user select a diagram, chooses ”Delete” option. 2. System shows a warning message. 3. The user confirms to delete. 4. System deletes the diagram and shows the result message.
Alternative flows:	• 4a The diagram is still shared to other users. – 4a1 System show error message and provide ”provoke all permission” option. – 4a2 The user confirm. – 4a3 System delete all of share permission. – 4a4 Use case continue with step 4.
Exception:	• 4a The user is not the owner of the diagram. – 4a1 System show error message. – 4a2 Use case stops.

Table 5.10: Use case: Delete Diagram.

Use case ID:	UC-12
Use Case Name:	Logout.
Triggering Event:	The user is done using the app and want to logout.
Brief Description:	The user chooses ”Log out” and ends their logging session.
Actors:	The user
Preconditions:	• The user is logged in.
Post-conditions:	• The user is logged out.
Normal flows:	1. The user choose ”Option” and ”Logout”. 2. System logs user out and invalidates the API key. 3. System redirects to login screen.
Alternative flows:	N/A
Exception:	N/A

Table 5.11: Use case: Logout.

Chapter 6

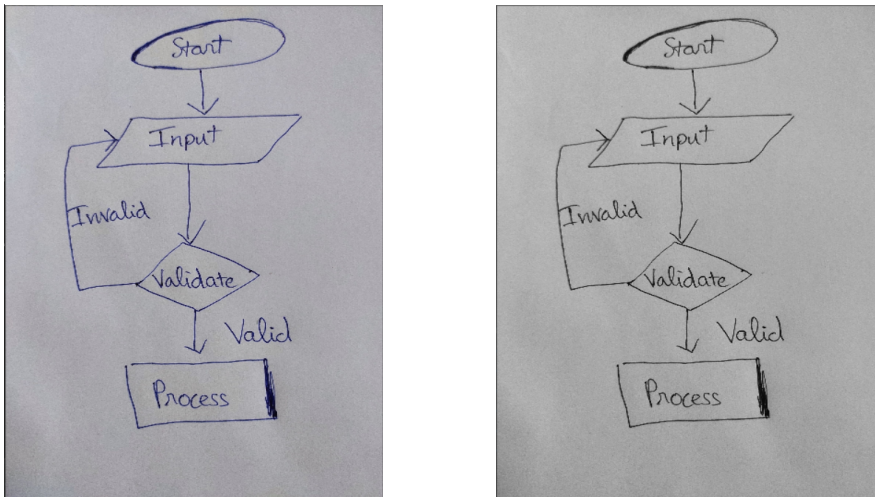
Initial experiments

6.1 Preprocessing experiment

In this experiment about preprocessing, we mainly use the library OpenCV to provide the results for each techniques mentioned in Chapter 3 and give an evaluation for the experiment result.
In this section, we are going through each steps in the process. We also have some pictures to make it clearer about the result of each step. The original image we used in this section is captured in a good environment condition.

6.1.1 Convert RGB image to grayscale image

The first step of all is convert a RGB image into grayscale image. This is a simple to remove the color part from the captured image. In this step, we use luminosity method and the result can see in the 6.1b:



(a) Original Image (b) Grayscale image

Figure 6.1: Figure 6.1a is the original handwriting flow chart; figure 6.1b is grayscale version of the figure 6.1a

6.1.2 Histogram equalization

In this step, we use Histogram Equalization technique to improve contrasts in the grayscale image.

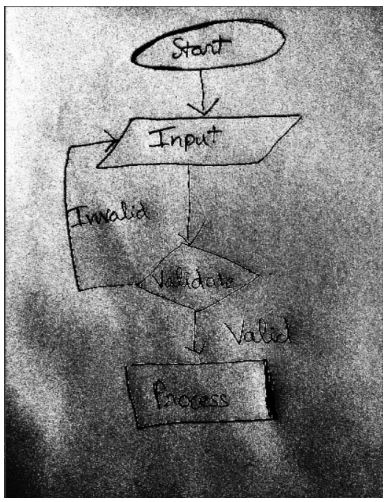
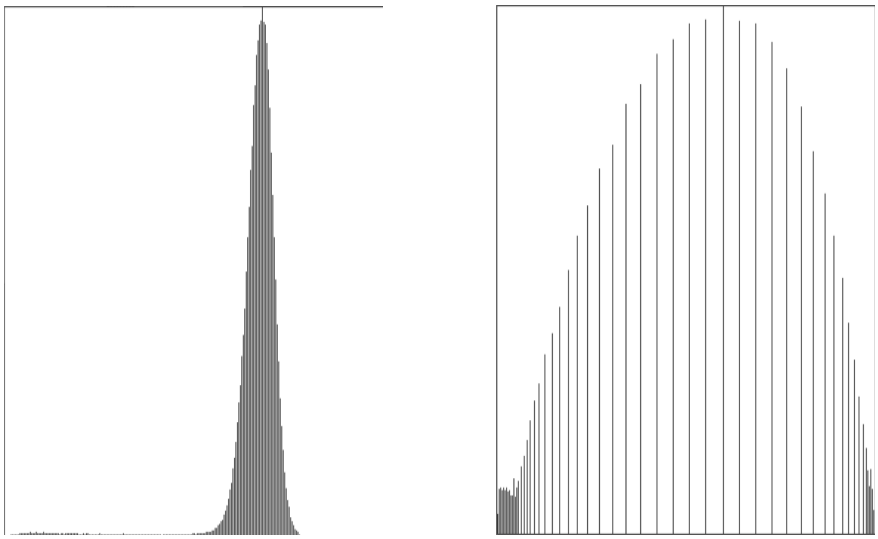


Figure 6.2: Histogram equalized image converted from figure 6.1b



(a) Original Histogram (b) Equalized histogram

Figure 6.3: Figure 6.3a is the histogram of figure 6.1b; figure 6.3b is the figure 6.3a converted to by using Histogram Equalization

Observe the figure 6.2, we can see that this procedure make the image worse than the figure 6.1b. From the result, we decided to remove the step Histogram Equalization from preprocessing.

6.1.3 Image smoothing

In this step, we filter the noise from the image. As we can see from the figure 6.1 (b), the noise in the figure is Gaussian noise, which represents statistical noise having probability density function equal to the normal distribution, which is also known as the Gaussian distribution.

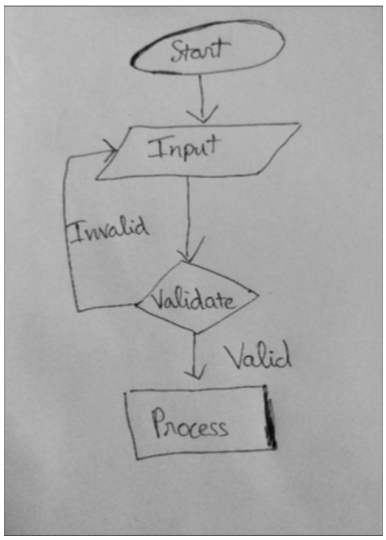


Figure 6.4: Gaussian blur applies on 6.1b

Therefore, to remove this type of noise, Gaussian filter is used in this case. The blurred image after using Gaussian filter is in figure 6.4.

6.1.4 Convert into binary image

In this step, we use the Adaptive Thresholding technique to binarize figure 6.4. The result of this step is in figure 6.5. In addition, we also show the result of converting the histogram equalization image, which also is blurred by Gaussian blur, into binary image in this figure.

From figure 6.5, it can be seen that there are speckle noises which is needed to be removed.

6.1.5 Speckle filter

This is the final step of preprocessing. In this step, we use the filterSpeckles() function from OpenCV to remove the blobs in the binary image. The result we get is in figure 6.6.

This function gives a good result with the figure 6.6, which separates the flowchart with the black color from the white background.

6.2 Web server communication

This system is designed to transfer images taken from client devices and send them to the web server to process. However, most of the current mobile device has a great camera quality which can produce high definition pictures but also has a large size. Trying to transfer the whole file in one request may cause disruption and corruption of the file. However, the Hypertext Transfer Protocol (HTTP) supports a Multi part Request, which is a combination of one or more sets of data into a single body. The body is then divided into many ”body parts”, each of them has an encapsulation boundary ahead in the combination and the last one is enclosed with a closing boundary [52]. The content-type ”multipart/form-data” allows the app to send the image by blocks of data, which will then be collected by the web server.

Flutter HTTP package supports multipart/form-data requests. When this request is called, it automatically sets the Content-Type header to multipart/form-data. This value will override

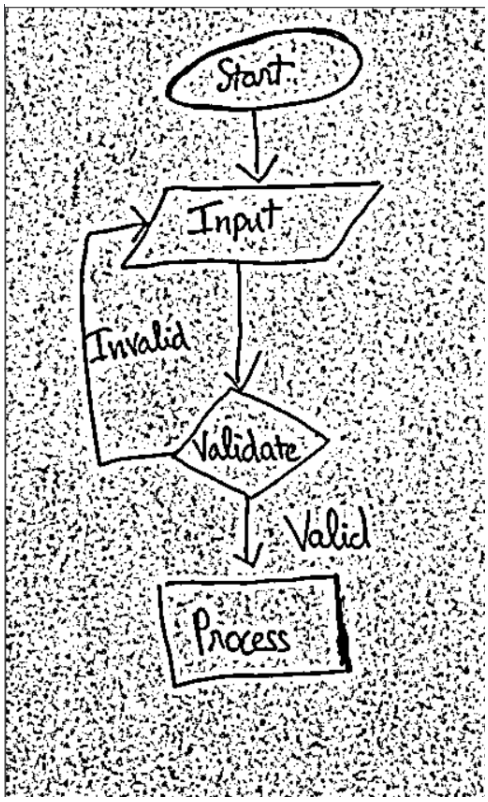


Figure 6.5: Binary image

any value set by the user. There are many ways to perform this request, in this experiment, we used MultipartFile.fromBytes, which will create a new multi part file from a byte array. That means first the image needs to be converted into a list of bytes, then will be transformed into a multipart file and send over (see Figure 6.7). The web server will receive the file as a stream (see Figure 6.8).

6.3 Display diagram on device

6.3.1 Interactive Viewer and Matrix4

In order to handle user action like panning, zooming and moving around the diagram, the surface that used to draw the diagram on need to be interactive. One of the most powerful widget (all components in Flutter is fundamentally defined as a widget) in Flutter is the transform class widget which is able to change the perspective of the user of a widget. Basically it makes changes in how the widget look and behave so that developer can create new and more complex component from the existed one. It is controlled by a four dimensions matrix that define how the widget will transform called ”matrix4”. While Flutter also provides other easier ways for scaling, rotating and translating, ”matrix4” allows more customization. ”Interactive Viewer” is one of the widget created from the transform class that allows user to transform all the components inside by dragging to pan or pinching to zoom. The difference of it from the rest is it can create a wide space for interacting and is mainly used to grab the widgets that is too big to fit inside the screen, which is a perfect space to draw the diagram on. ”Interactive Viewer” transforms itself by changing some specific value in the ”matrix4”.

For example, in the table 6.9, we can see a 4x4 matrix represent the ”matrix4”, by default

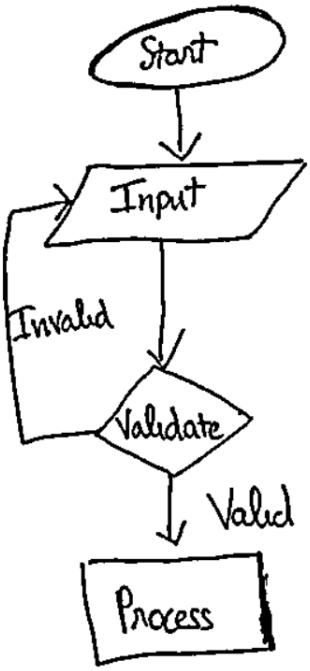


Figure 6.6: Filter speckles in binary image

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL
D/FilePickerDelegate(13606): File path:[com.mr.flutter.plugin.filepicker.FileInfo@8936e98]
I/flutter (13606): Converting image to bytes
I/flutter (13606): [255, 216, 255, 225, 2, 112, 69, 120, 105, 102, 0, 0, 77, 77, 0, 42, 0, 0, 0,
0, 0, 1, 0, 0, 5, 0, 1, 50, 0, 2, 0, 0, 0, 20, 0, 0, 0, 130, 1, 18, 0, 3, 0, 0, 0, 1, 0, 1, 0, 0
0, 0, 7, 0, 0, 0, 150, 0, 0, 0, 0, 65, 79, 83, 80, 32, 111, 110, 32, 73, 65, 32, 69, 109, 117, 1
0, 71, 111, 111, 103, 108, 101, 0, 0, 16, 130, 157, 0, 5, 0, 0, 0, 1, 0, 0, 1, 99, 130, 154, 0,
4, 49, 53, 0, 146, 144, 0, 2, 0, 0, 0, 4, 54, 49, 53, 0, 146, 10, 0, 5, 0, 0, 0, 1, 0, 0, 1, 115
0, 20, 0, 0, 1, 123, 144, 3, 0, 2, 0, 0, 0, 20, 0, 0,
I/flutter (13606): Sending
I/flutter (13606): Sent
```

Figure 6.7: Client application sending image log

```
PS E:\Coding\NodeJS\Testing> npm run start

> test_restful@1.0.0 start E:\Coding\NodeJS\Testing
> nodemon server.js

[nodemon] 2.0.6
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
RESTful API server started on: 3000
Receiving a file fromuploads\1258ffc7b3863f55638b5d27fd57a0cb
```

Figure 6.8: Server log

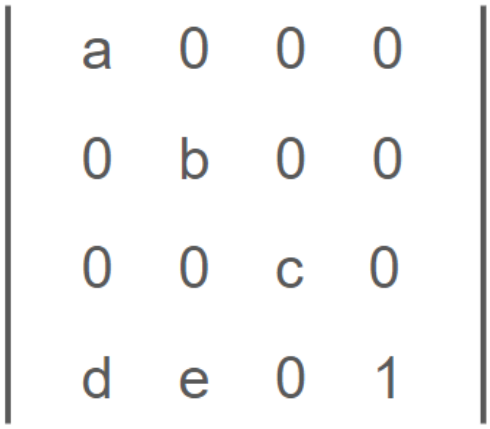


Figure 6.9: Example of matrix4

it is an identity matrix (which has all the value in the diagonal line equal to 1). In that state, the widget is in its normal size (see Figure 6.10). Then it can be tweeted in order to achieve the prefer shape.

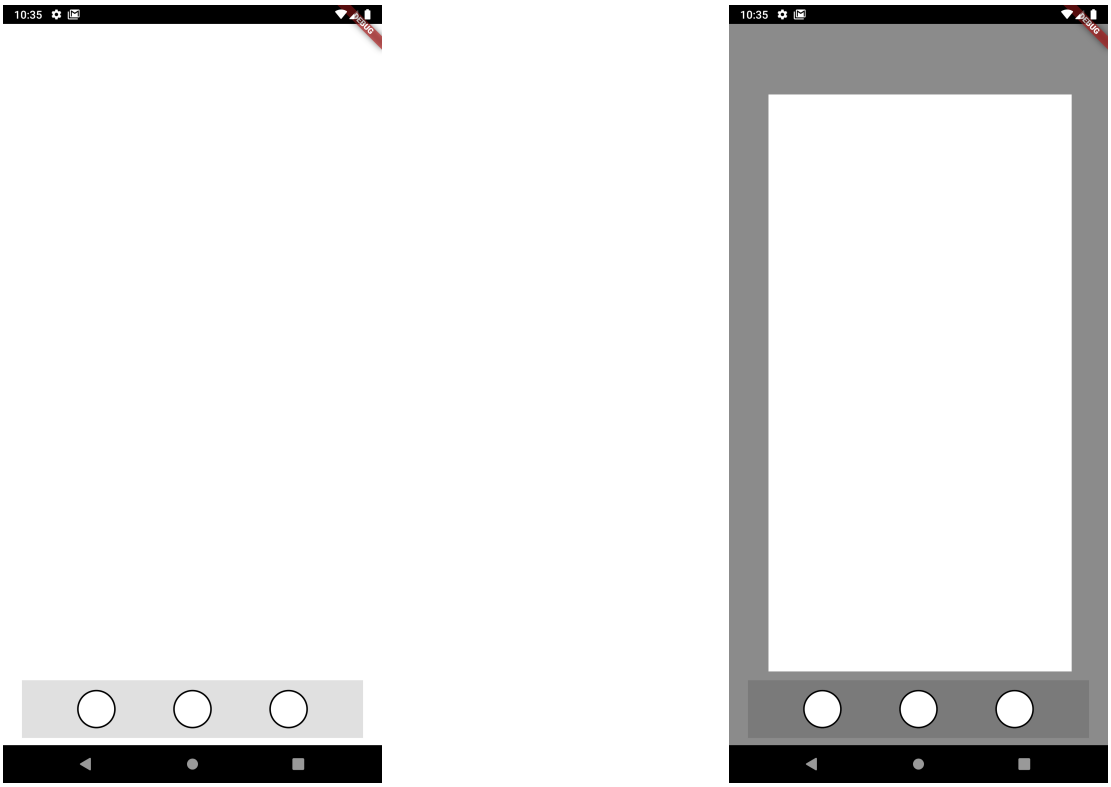
To scale the widget (change its size to make it bigger or smaller): we can change the value ”a” and ”b” in the table 6.9 into a value greater than zero (zero value will make the matrix unable to reverted). Value smaller than one will make it shrink and value greater than one will expand it in the corresponding dimension. In the table, value ”a” controls the X-dimension or the width of the space (see Figure 6.11a) and ”b” does the same for the height (Y-dimension) (see Figure 6.11b).

In a two dimension perspective, that maybe enough to express most of the form of the space. However, there is also the value ”c” that may influence the way user interact with the space, which scaling in the Z-dimension. For instance, in Figure 6.12a, value ”c” in position 3,3 has been changed to 2.0 but the space remain unchanged. In theory, the Z-dimension represents how deep the widget should look like, then it may seem to be unnecessary in a 2D space, but as it is zoomed out (see Figure 6.12b) then the zooming scale has been increased, as if the widget is now further from the screen perspective.

Finally, The space can be moved the space around in X and Y dimension by changing value in ”d” (4,1) and ”e” (4,2) position. Value in this position will be consider as the amount of pixels that the space move. For example, in Figure 6.13a, ”d” is set to 100, that means the space will move right 100 pixels. The same thing with Figure 6.13b with ”e” equals 100, the space will move down 100 pixels.

6.3.2 Result

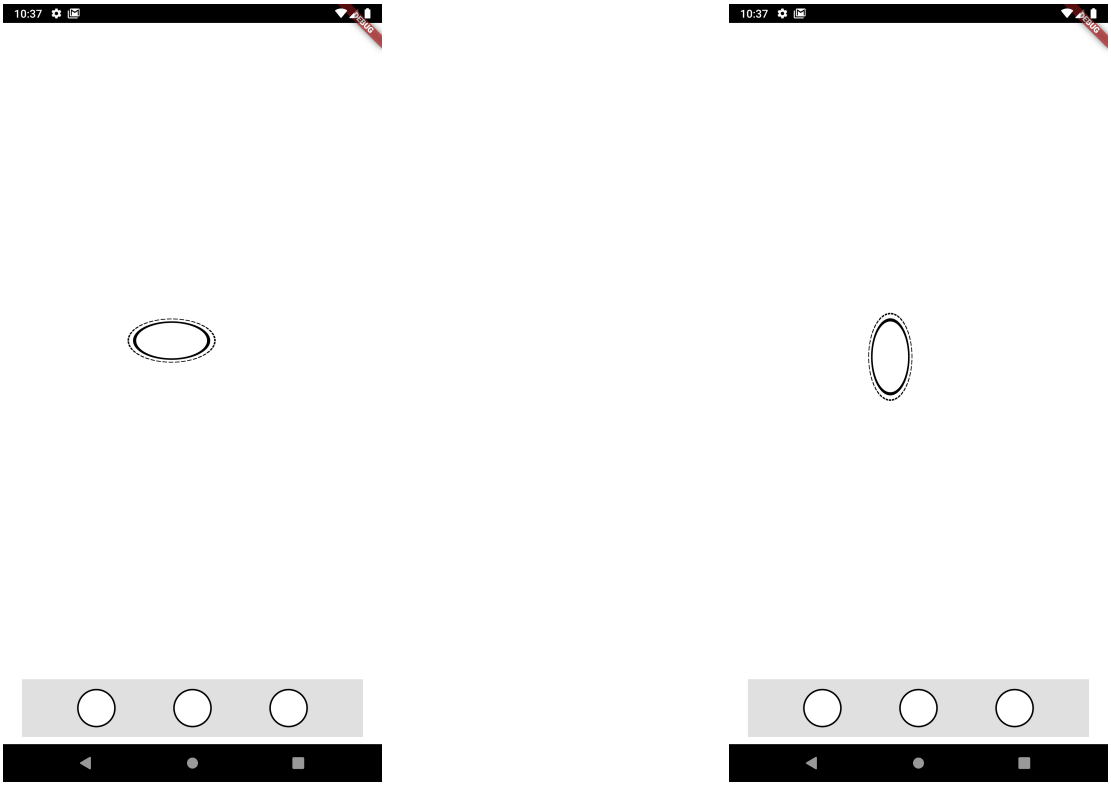
After applying the previous widget and display the symbol and line with a stack in the diagram space, we get the result as show in figure 6.14.



(a) Default

(b) Zoom out from the default

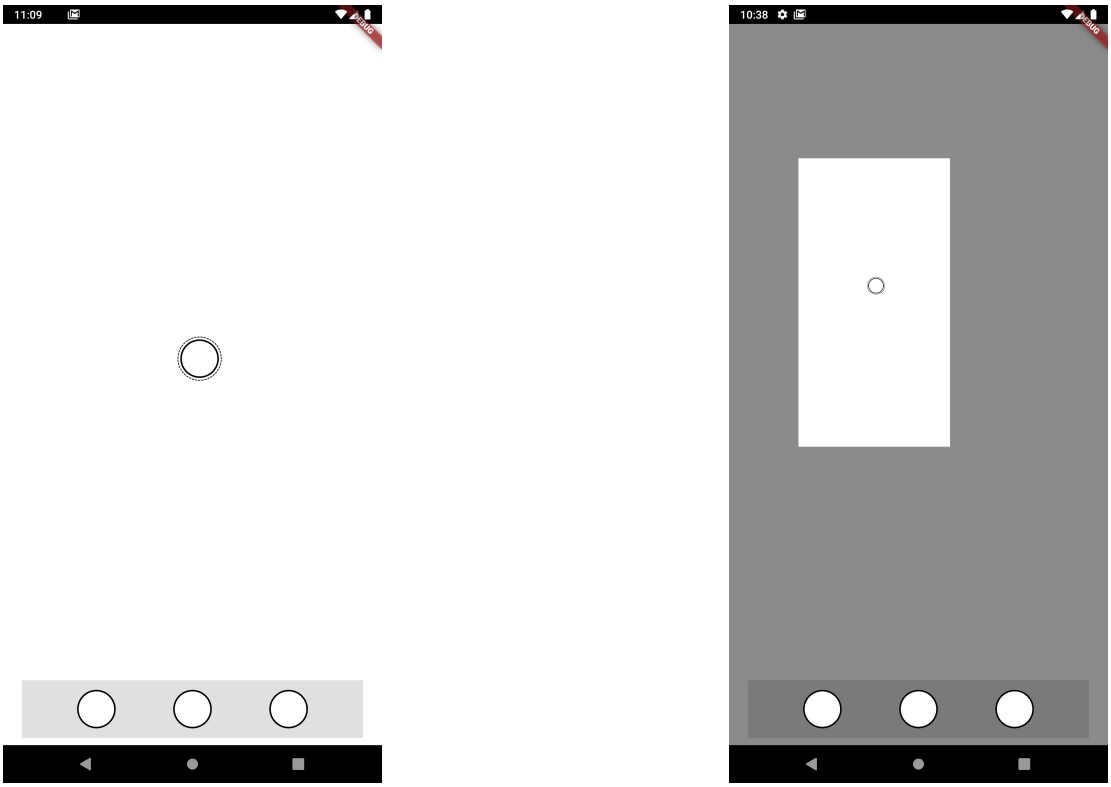
Figure 6.10: Interactive space using identity matrix



(a) Value a (1,1) = 2.0

(b) Value b (2,2) = 2.0

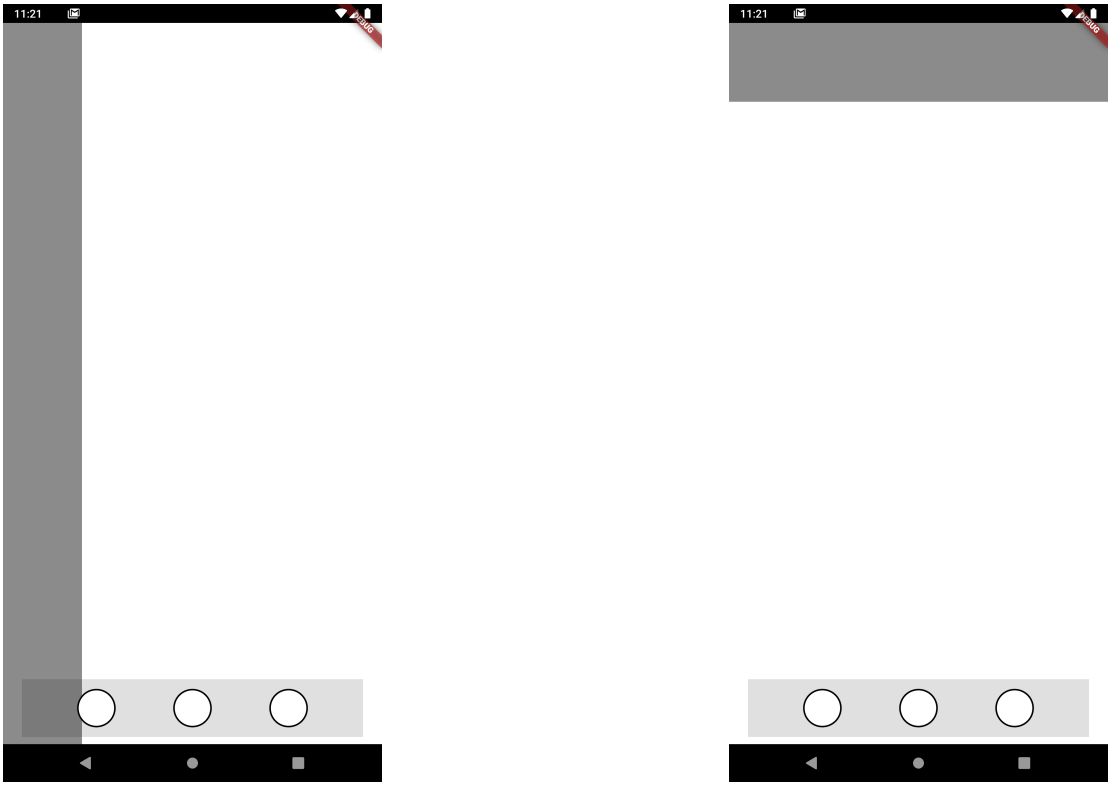
Figure 6.11: Scaling in X and Y



(a) Value c (3,3) = 2.0

(b) Zoom out with c = 2.0

Figure 6.12: Scaling in Z



(a) Value d (4,1) = 100

(b) Value e (4,2) = 100

Figure 6.13: Moving the space

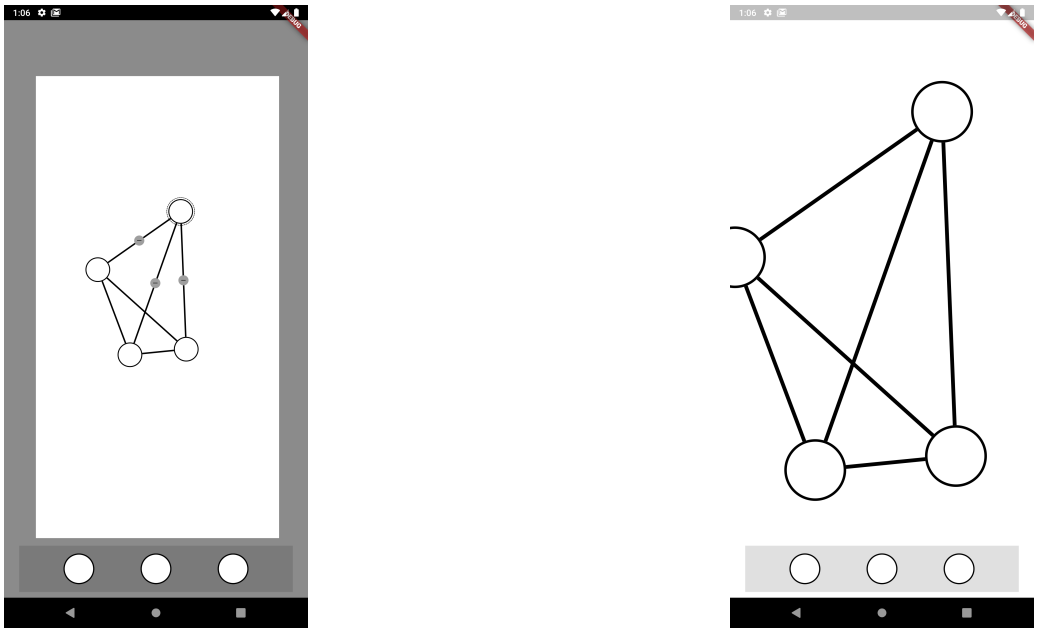


Figure 6.14: Diagram display result

Chapter 7

Conclusion and Future Work

7.1 conclusion

The field diagram detection is an interesting and complex study domain. Although there are many research about on-line diagram, there is still less research about recognizing off-line diagram. With this thesis proposal, we have achieved new background knowledge about the computer vision. We are able to access each procedures in image processing which is also a new field for us. We finds many approaches for handwriting flowchart recognition and we select the suitable approaches using Faster R-CNN, YOLO v4 and RefineDet.
During the research process, our group still had many problems in teamwork, lack of time management skills which affects negatively our result. This problems may become a wall in our work in the future, which we need to overcome and improve our working process.

7.2 Challenges

- One of the biggest problem with server-client system is scaling or the amount of customers serving at the same time. Most of our experiment is conducted in the local machine so that there is a need to find a way to provide more stable service in the final product.
- The target is to support a wide range of products so as many devices can install this app as possible. Then it can lead to performance issues and the need to find a balance between stable functioning and feature variety.
- Security is also important since many of these data can be very crucial. All of the information sent or receive by the app and the server need to be secured in order to prevent data leak.

7.3 Future work

In the future, we are going to implement the flowchart recognition system and also evaluate the capabilities of the programming languages and the support framework to reduce the constraints of our proposed system while solving all of the listed challenges.

References

[1] Ajay Kumar Boyat and Brijendra Kumar Joshi. A review paper: Noise models in digital image processing. *International Journal of Science and Research*, pages 1–11, 2015.

[2] Abdalla Mohamed Hambal, Dr. Zhijun Pei, and Faustini Libent Ishabailu. Image noise reduction and filtering techniques. *International Journal of Science and Research*, pages 1–5, 2017.

[3] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition*, 1:511–518, 2001.

[4] P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57:137–154, 2004.

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition*, 1:886–893, 2005.

[6] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Cascade object detection with deformable part models. *Computer Vision and Pattern Recognition*, page 2241–2248, 2010.

[7] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. *International Conference on Computer Vision*, page 89–96, 2011.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, page 1097–1105, 2012.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffne. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, pages 2278–2324, 1998.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 2012.

[11] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv:1409.4842*, 2014.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *arXiv:1512.00567*, 2015.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *Computer Vision and Pattern Recognition*, 2016.

[15] L. Deng, M. L. Seltzer, D. Yu, A. Acero, A. r. Mohamed, and G. Hinton. Binary coding of speech spectrograms using a deep auto-encoder. *INTERSPEECH*, 2010.

[16] G. E. Dahl, M. A. Ranzato, A. r. Mohamed, and G. Hinton. Phone recognition with the mean-covariance restricted boltzmann machine. *Neural Information Processing Systems*, 2010.

[17] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing coadaptation of feature detectors. *arXiv:1207.0580*, 2012.

[18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, 2015.

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *Computer Society Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Regionbased convolutional networks for accurate object detection and segmentation. *Transactions on Pattern Analysis and Machine Intelligence*, 38:142–158, 2016.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *arXiv:1406.4729*, 2014.

[22] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *Computer Vision and Pattern Recognition*, 2006.

[23] F. Perronnin, J. Sanchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. *European Conference on Computer Vision*, 2010.

[24] R. Girshick. Fast r-cnn. *International Conference on Computer Vision*, 2015.

[25] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv:1506.01497*, 2015.

[26] Y. Li, J. Dai, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. *arXiv:1605.06409*, 2016.

[27] T.-Y. Lin, P. Dollar, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. *arXiv:1612.03144*, 2017.

[28] K. He, G. Gkioxari, P. Dollar, and R. B. Girshick. Mask r-cnn. *International Conference on Computer Vision*, 2017.

[29] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *arXiv:1506.02640*, 2015.

[30] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *arXiv:1612.08242*, 2016.

[31] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv:1804.02767*, 2018.

[32] A. Bochkovskiy and H. Y. M. Liao C. Y. Wang. Yolov4: Optimal speed and accuracy of object detection. *arXiv:2004.10934*, 2020.

[33] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. *European Conference on Computer Vision*, 2016.

[34] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal loss for dense object detection. *Pattern Analysis and Machine Intelligence*, 2018.

[35] Zhang. Shifeng, Wen. Longyin, Bian. Xiao, Lei. Zhen, and Li. Stan Z. Single-shot refinement neural network for object detection. In *CVPR*, 2018.

[36] J.-P. Valois, M. Cote, and M. Cheriet. Online recognition of sketched electrical diagrams. *Proceedings of Sixth International Conference on Document Analysis and Recognition*, 2001.

[37] G. Feng, C. Viard-Gaudin, and Z. Sun. On-line hand-drawn electric circuit diagram recognition using 2d dynamic programming. *Pattern Recognition*, page 3215–3223, 2009.

[38] Ouyang T.Y. and Davis R. Chemink: A natural real-time recognition system for chemical drawings. *International Conference on Intelligent User Interfaces*, page 267–276, 2011.

[39] Qi Y., Szummer M., and Minka T.P. Diagram structure recognition by bayesian conditional random fields. *Conference on Computer Vision and Pattern Recognition*, page 191–196, 2005.

[40] A. Lemaitre, H. Mouchere, J. Camillerapp, and B. Couasnon. Interest of syntactic knowledge for on-line flowchart recognition. *Graphics Recognition. New Trends and Challenges*, page 89–98, 2013.

[41] C. Wang, H. Mouchere, C. Viard-Gaudin, and L. Jin. Combined segmentation and recognition of online handwritten diagrams with high order markov random field. *International Conference on Frontiers in Handwriting Recognition*, page 252–257, 2016.

[42] C. Wang, H. Mouchere, and C. Viard-Gaudin. Online flowchart understanding by combining max-margin markov random field with grammatical analysis. *International Journal on Document Analysis and Recognition*, page 123–136, 2017.

[43] M. Bresler, D. Prusa, and V. Hlavac. Modeling flowchart structure recognition as a max-sum problem. *International Conference on Document Analysis and Recognition*, page 1215–1219, 2013.

[44] M. Bresler, D. Prusa, and V. Hlavac. Online recognition of sketched arrow-connected diagrams. *International Journal on Document Analysis and Recognition*, page 253–267, 2016.

[45] M. Bresler, D. Prusa, and V. Hlavac. Recognizing off-line flowcharts by reconstructing strokes and using on-line recognition techniques. *International Conference on Frontiers in Handwriting Recognition*, page 48–53, 2016.

[46] A. Bhattacharya, S. Roy, N. Sarkar, and S. Malakar. Circuit component detection in offline hand drawn electrical and electronic circuit diagram. *Calcutta Conference*, page 151–156, 2020.

[47] Manoj Sonkusare and Narendra Sahu. A survey on handwritten character recognition (hcr) techniques for english alphabets. *Advances in Vision Computing: An International Journal*, pages 1–11, 2016.

[48] Anil. K. Jain and Torfinn Taxt. Feature extraction methods for character recognition-a survey. *Pattern Recognition*, pages 641–662, 1996.

[49] Tony Flores. Median filtering with python and opencv, 2019. Last accessed 25 December 2020.

[50] Flutter.

[51] Nodejs.

[52] The multipart content-type.