Ge’ez Handwritten Character Recognition System with Machine Learning

A PROJECT REPORT

Submitted by

Name	ID
Eba Alemayehu	TER_0075_09
Kerebish Genet	TER_0083_09
Nardos Sharifo	TER_0088_09

in partial fulfilment for the award of the degree of

BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY

Under the guidance of

Mr. Baye Atinafu

ADVISOR SIGNATURE

DEPARTMENT OF SOFTWARE ENGINEERING

INSTITUTE OF TECHNOLOGY

DEBRE MARKOS UNIVERSITY

DEBRE MARKOS

February 2020

Abstract

The project tries to build a comprehensive handwritten Ge’ez characters dataset. In the project we try to find the best deep learning model witch best fit to learn handwritten Ge’ez characters. A couple of sub systems are also built as a part of the project.

Acknowledgment

We sincerely want to thank our adviser who were helping us in this journey. We also want to thank our department for providing us an office support we have needed for our project. Last but not the list we want to thank all volunteers who participated in filling the quaternary.

Table of Contents

Abstract I

Acknowledgment II

List of table 3

List of figures .... **Error! Bookmark not defined.**

1. Introduction 6

1.1. Background of the project 6

1.2. Statement of the problem 7

1.3. Literature review 7

1.4. The Objective of project 9

1.4.1. General objective 9

1.4.2. Specific objective 9

1.5. Scope of the project 9

1.6. Significance of the project 9

1.7. Tools and methodology 9

1.7.1. Data Collection methodology 9

1.7.2. Technologies to be used 10

1.7.3. System requirements hardware and software 10

1.7.4. System modeling tools 10

1.8. Feasibility study 11

1.8.4. Technical 11

1.8.5. Operational 11

1.8.6. Economical 11

2. System Analysis 11

2.7. Overview of existing system 11

2.8. System Requirement Specification 11

2.8.4. Functional Requirements 11

2.8.5. Non-Functional Requirements 12

2.8.6. Overview of proposed system 12

3. Dataset Collection 13

3.7. Data gathering methodology 13

3.8. Sampling mechanism 13

3.9. Data analysis and preprocessing 13

3.10. Software Requirement specification for data collection app 22

3.10.4. Functional requirement 22

3.10.5. Non-functional requirement 22

3.10.6. Overview of Proposed system 22

3.10.7. Business rules... 22

3.11. System requirement analysis for data collection application 23

3.11.1Actor and use case identification 23

3.11.2. Use cases 23

3.11.3. Sequence Diagram 28

3.11.4. Activity Diagram 31

3.11.5. Analysis Class Diagram 32

3.12. System Design 32

3.12.1. Design Class Diagram... 32

3.12.2. Physical Data Model 33

3.12.3. User Interface Design 33

3.12.4. Deployment Design 35

4. Implementation 35

4.1. Overview of the programming language used 35

4.2. Algorithms Used 36

4.3. Sample Codes 38

4.3.1. Image Slicer 38

4.3.2. Extract characters 45

4.3.3. Training the model 51

5. Testings...**Error! Bookmark not defined.**

5.1. Sample Test ....**Error! Bookmark not defined.**

6. Conclusion and Recommendations 65

6.1. Conclusion 65

6.2. Recommendation and Future Enhancement 65

6.2.1. Transfer leaning 65

6.2.2. Multitask learning 65

Appendices 67

References 71

List of table

Table 1: Beget plan table.. **Error! Bookmark not defined.**
Table 2: Actors and their description 23
Table 3: List of use cases.. 23
Table 4: Login use case description 24
Table 5: Logout use case description 24
Table 6: Upload questioner use case description 25
Table 7: Label images use case description 26
Table 8: Create agent account use case description... 26
Table 9: Revoke agent access use case description 27

List of table

Figure 1: Amharic scripts 7
Figure 2: Original scanned image 14
Figure 3: Resized gray scale image 15
Figure 4: Filtering and OTSU threshold applied to the image 16
Figure 5: Morphological adjustment is made on the image 17
Figure 6: Horizontal lines are extracted from image 18
Figure 7: Vertical lines are extracted from image 18
Figure 8: Bitwise or are applied on extracted horizontal lines and extracted vertical lines 18
Figure 9: Canny edge detection is applied on the image on figure 7 19
Figure 10: Sample segmented images 20
Figure 11: Algorithm for sorting bounding box coordinates Figure 12: Algorithm for segmenting images in the questioner 21
Figure 13: Use case diagram for data collection application 23
Figure 14: Sequence diagram for login 28
Figure 15: Logout sequence diagram 29
Figure 16: Create user sequence diagram 29
Figure 17: Label images sequence diagram 30
Figure 18: Upload questioner sequence diagram 30
Figure 19: Figure 16: Label image sequence diagram 31
Figure 20:Analysis class diagram 32
Figure 21: Design class diagram 33
Figure 22: Physical data model 33
Figure 23: Login interface 33
Figure 24: Upload questioner form 34
Figure 25: Questioner images list 34
Figure 26: Questionnaires list interface design 35
Figure 27: Display design diagram 35
Figure 28 Slicing algorithms flow chart 36
Figure 29: Sort coordinates algorithms flow chart 37
Figure 30 39
Figure 31 39
Figure 32 40
Figure 33 41
Figure 34 41
Figure 35 43
Figure 36 43
Figure 37 45
Figure 38 46
Figure 39 46
Figure 40 47

Figure 41 47
Figure 42 48
Figure 43 48
Figure 44 49
Figure 45 50
Figure 46 63
Figure 47 64
Figure 48 65
Figure 49: Questioner first page 67
Figure 50: Questioner page 2 a 68
Figure 51:Questioner page 2b 69
Figure 52: Questioner page 2 c 70

1. Introduction

1.1. Background of the project

Ge’ez is liturgical language of the Ethiopian church. Ge’ez is a Semitic language of the Southern Peripheral group, to which also belong the South Arabic dialects and Amharic, one of the principal languages of Ethiopia ((1)). Ge’ez has its own writing style and alphabet. Both Ge’ez and the related languages of Ethiopia are written and read from left to right, in contrast to the other Semitic languages. Extinct as a vernacular language, Ge’ez is the ancestor of the modern Tigrinya and Tigré languages of Eritrea and Ethiopia. The oldest known inscription in the language dates from the 3rd or 4th century and is written in a script that does not indicate vowels.

Most of Ethiopian history and documentations were written in Ge’ez characters. Either with Ge’ez language itself or other descendent languages of Ge’ez like Amharic. Before the era of computers and automation the government and other organizations used handwritten documents. These documents contain accumulated wisdom of our forefathers, the history of our country, the history of gov’t records like police records etc... Therefore, we need some automated way to help us digitize these documents in order to make storage and distribution of these documents match easier.

ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ	
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ	ሏ
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ	ሗ
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ	ሟ
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ	ሧ
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ	ሯ
ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ	ሷ
ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ	ሿ
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ	ቋ
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ	ቧ
ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ	ቯ
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ	ቷ
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ	ቿ
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ	ኋ
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ	ኗ
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ	ኟ
አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ	ኧ

ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ	
ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ	
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ	ዟ
ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ	ዧ
የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ	
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ	ዷ
ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ	ጇ
ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ	ጓ
ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ	ጧ
ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ	ጯ
ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ	ጷ
ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ	ጿ
ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ	
ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ	ፏ
ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ	ፗ
፡	፣	፤	፥	።	፩	፪	፫
፬	፭	፮	፯	፰	፱	፲	፳

ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ	ኳ
ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ	ዃ

፴	፵	፶	፷	፸	፹	፺	፻

Figure 1: Amharic scripts

1.2. Statement of the problem

Typing Ge’ez characters into computer is relatively harder because the standard QWERTY keyboard which most of us use is not designed for Amharic language. It also requires a lot of labor. It is a very time-consuming task. These days people tend to use mobile or tablet devices rather than the conventional desktop computers. These devices have no convenient way of writing a long text. Especially writing Amharic text is much more difficult because though there exist few applications which allow as to write Ge’ez there is no standard for the key layout.

On the other hand, artificial intelligence and fully intelligent systems are growing. These systems are expected to dominate the world. This kind of systems incorporate artificial general intelligence which means they almost mimic a human mind. One of the issues in the part of this big general system call machine learning is the issue of diversity. Machine learning is an algorithm which takes data and tries to learn from that data. So that if it will not be trained with our language we will be left over in the digital divide. Which negatively affect us and our language. Therefore, we need to develop AI systems which can mimic our culture and our language. Reading or recognizing character are one of the skills of language we need to train machines.

Humans can easily recognize characters one they have learned them in spite of how distorted they are but identifying handwritten characters is not an easy computer vision task in a traditional way of programming, because it would be impossible to be able to write rules about how each character are represented. Humans write characters in unpredictable way; style of writing differs from person to person. Therefore, we need to use another approach which is machine learning. This pause a big computer since challenge but recently machine learning had become good at this kind of computer vision tasks because know we have good enough computational power to teach computers to detect this unpredictable writing of characters.

1.3. Literature review

Though the growth of computer technology people still use handwriting as it is still easy and convenient way of writing. For computers it is very easy to process encoded text in other hand humans have a very good visual capability to process handwritten text. Researches has been trying to recognize both hand written and computer printed characters from a photo graph for a long time.

There are two peridium’s in handwritten recognition and segmentation-based and holistic segmentation base approach segment word image in to constituent character where are holistic approach tries to recognize the whole word ignoring character segmentation. Holistic approach also extracts repetitive features of the whole word and it is more pragmatic in case of cursive handwriting where characters are physically connected each other by segmentation turns out to be impractical (Yaregal Assabie a, 2019). Ge’ez writing doesn’t have a cursive writing characteristic which makes it easy for segmentation. There for segmentation-based recognition would be feasible. This research also uses this segmentation-based recognition (Birhanu, 2010).

Offline recognition of Latin, Chinese, Japanese, Indian and Arabic hand written has long area of active research. However Ethiopic handwriting recognition in general and Amharic word recognition in particular, is one of the least investigated problems (Yaregal Assabie a, 2019).

A couple of studies were made on Amharic, Ge’ez or in general Ethiopic character recognition. Some of them used machine learning techniques where as others used different tadeonal algorithms. As far as our knowledge one of the pioneers are the study by [John and Fiaz Hussain]. This paper describes the Amharic script and discusses the difficulties of applying conventional structural and syntactic recognition processes. Two statistical algorithms for identifying Amharic characters are described. In both, the characters are normalised for both size and orientation. The first compares the character against a series of templates. The secondderives a characteristic signature from the character and compares this against a set of signature templates (Cowell, 2003). Even though this algorthm prformed relatively well on computer printed characers with a certain font. It has avery poor performance in handwritten characater recogntion. Other studies have pusshed forward the research area with different techniques like HMMs, support vector machine (Yaregal Assabie a, 2019) (Gauri Katiyar1, 2017).

One of difficulties in recognizing amharic charaters are the number of unique characates and the simmilarty between them (Yaregal Assabie a, 2019). CNN have shown a remarkabel performance in recognizing such a commplex patterns (Sen Maitra, 2015). Recent remarkable research used estate of the art deep nural network and convolutional nural networks (Mesay Samuel Gondere, 2019). This research used CNN with the techinque called Multi task learning. The dataset used by this research were from the work of (Assabie & Bigun, 2008). It uses 12 unique handwritten characers and it use different kinds of agumantaion to increase the number of data to 450, 800 and 400 Traing, validation and test set. Finally the result of the reserch shows. Different accurecy lavels in average bettwen 50% and 75%.

1.4. The Objective of project

1.4.1. General objective

To build a computer system that can recognize any handwritten Ge’ez characters.

1.4.2. Specific objective

The specific objective of this project is:

- Preparing Ge’ez characters dataset which is publicly available and anyone who want to experiment on it can try out.
- To find the best learning algorithm and neural network architecture
- To train a model which can classify Amharic characters
- To make application software which make use of the trained model

1.5. Scope of the project

This project consists of:

- Ge’ez character dataset collection
- Designing the learning algorithm and architecture
- Training the model with the collected dataset and Designed architecture
- Building an application on the top of the trained model
- Different small programs that help to automate some tasks on the process for instance data collection.

This project does not include:

- Any natural language processing. Our system does not understand the meaning of a text.
- No semantic analysis or data mining on text is done.
- Word or sentence-based recognition. Our system is character-based recognition.

1.6. Significance of the project

The significance of the project can be seen in different dimensions. On one side the system we are going to build an application which can solve some problems. It includes API that developers we amazing idea can build applications. On the other side when we see the big picture it could be one step forward for next AI projects that can be done with our juniors.

1.7. Tools and methodology

1.7.1. Data Collection methodology

To train our model we need a lot of data. We are planning two ways to collect data.

I. By distributing questionnaire paper to different people to get there handwriting.

II. Building an android app which help us collect characters data. The app will have a canvas to enable draw characters on screen.

1.7.2. Technologies to be used

I. Programming languages
  - Python
  - JavaScript
  - Java, swift or dart (optional)
  - Html and CSS

II. Tools and technologies
  - OpenCV
  - Tensor flow
  - Flutter
  - Koras
  - NumPy
  - Matplotlib
  - Django or flask

1.7.3. System requirements hardware and software

2. Operating system
- Linux: will used as the development and training operating system
- Windows: for documentation and some drawing
- MacOS: will be used for compiling iOS apps
3. Software
- Android studio for android development
- Visual studio code as a text editor
- Apache or nginx serves
- Google chrome for testing and debugging JavaScript
4. Hardware
- Two computers one for training the model one for a development. The specification for these two computers are listed as follows.
  a) Training server: core i7 processor, 16GB RAM 1 TB storage with GPU capability.
  b) Development pc: core i7 processor, 8GB RAM 1 TB storage.
- Android and iPhone devices for testing
- Printed paper for data collection
- Other office apparatus for different purposes

1.7.4. System modeling tools

- Microsoft Visio and project
- StarUML as modeling tool

1.8. Feasibility study

1.8.4. Technical

Thanks to the big improvements that deep learning brings into the computer vision world. Deep learning technique can fit a very complex unstructured data we think the system is technically feasible with the resources we have now.

1.8.5. Operational

Through neural networks are relatively expensive in terms of computational resources we think it is feasible to operate our system in a computational power we have today. We can run our networks on cluster of computers on our lab or a cloud.

1.8.6. Economical

As we have mentioned earlier on the problem statement typing characters are time and resource consuming this project will allow our users to save a lot of time and resource so it is economically feasible.

2. System Analysis

2.7. Overview of existing system

Whenever we want to change a handwritten text in to computer understandable (editable text). We have to type the text again in to a computer system. People usually higher typists to do this work. In press industry usually, authors like writing in handwriting. When they are done with writing they take their work to the typists to type done to a computer. Also, media reporters take notes one some scenarios and type text to make news. We can mention a lot of areas where typing is used to change handwritten text to a computer.

Though there exist some character recognition system and a very few researches on handwritten Amharic character recognition systems, we don't think there still exist any Amharic handwritten character recognition system yet. Especially systems that are available for developers to work on more application systems which are based on this character recognition systems.

2.8. System Requirement Specification

2.8.4. Functional Requirements

- ✓ FR1: The system should allow data collection for Ge'ez handwritten characters
- ✓ FR2: The system give access to authorized users to label the images
- ✓ FR3: The system should segment characters from the questioner scanned image
- ✓ FR4: The system should be able to classify handwritten characters at minimum of >75% accuracy

- ✓ FR5: The system should detect, recognize and localize handwritten characters from scanned image or digital input

2.8.5. Non-Functional Requirements

1. Response Time:
   - ✓ TR1: The system should have a response time less than 10 seconds for recognition, detection and localization
2. Capacity
   - ✓ TR2: The system is expected to handle 100 recognition simultaneously
3. User interface:
   - ✓ TR3: The system should have a standard user interface that is easy to use
4. Authorization:
   - ✓ TR 4: only authorized users should label training datasets
5. Device and hardware:
   - ✓ TR 6: The system should run on mobile devices well as computer devices as a client
   - ✓ TR 7: The system should run on a system which has a GPU or TPU for faster training and neural network propagation at backend.
6. Operating systems:
   - ✓ TR 8: The system should run on android, IOS, Linux and windows Operating systems
   - ✓ TR 9: The system should run on any server operating system but Linux server is recommended.
7. Browser:
   - ✓ TR 10: the system should support Google chrome, chromium based operating systems, Mozilla Firefox, opera, safari and edge.

2.8.6. Overview of proposed system

The proposed system is a trained machine learning model which can recognize any handwritten character recognition. The system uses neural networks in order to achieve this task. The network will be a classifier network.

Ones we could train a model with a satisfactory accuracy level we will make different applications and interfaces which uses this model. This includes a web app, mobile app, desktop app and an API for developers. There are a lot of applications which can be built on this model. Some of them are:

- Amharic writing learning app which teaches how to write Amharic characters
- Amharic road signs reading and translation app
- Amharic optical character recognition systems

3. Dataset Collection

Machine learning algorithms inherently require a lot of data. The more the data the better the learning accuracy of the machine learning algorithm will be. More data also help as to easily overcome the bias variance trade of problem.

In our research we couldn’t find any comprehensive dataset for Ge’ez handwritten character recognition. Though some researches tried to prepare some dataset, their dataset is either not complete or the dataset is not available. There for we have prepared our own dataset which we will use in our model training in the next phase of the project. We will also make this dataset public and opensource so that any one interested in the Ge’ez handwritten character recognition use it.

We have also developed a system which help us automate the data collection and labeling task. Though human intervention is required, the system allow to upload scanned images of questioner and slice the handwritten characters. It also allows labeling of the characters.

3.7. Data gathering methodology

We have used questioner data collection method. We have prepared a questioner for people to fill in for us. Our questioner has 3 parts. The first part of our questioner instructs the person to rewrite an Amharic poem given on the questioner. We have prepared 6 different poems. Each volunteer participant will get one of the 6 poems. This part of the questioner allows us to test our trained model on consecutively written words than independently written characters. This will give us more real-world scenario. The second part of our questioner instruct the user to write 291 characters on the table provided. This are characters we are going to segment and train our model with. The third section will have the demographic data of the volunteer participant which include age, gender and educational level.

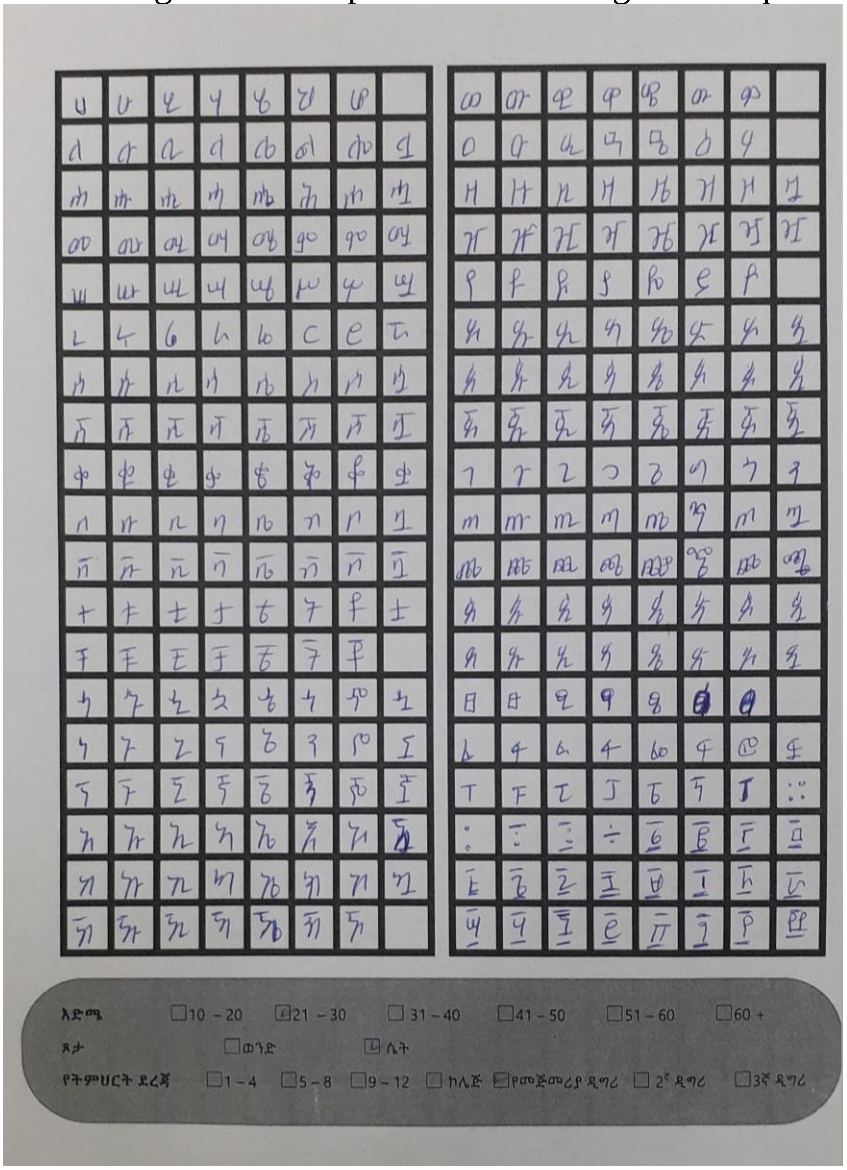
3.8. Sampling mechanism

Random sampling method was used to select our sample population. We have tried to collect the data from any volunteers because we need a lot of data. We could collect 485 unique data from 209 volunteer participants. In general, we have managed to collect 141,135 unique characters. This data collection will also continue one phase to of our project.

3.9. Data analysis and preprocessing

After we have collected the questioner, we have developed image processing algorithm which segment characters on the section two of questioner. The algorithm will slice each character using OpenCV image processing library and save each character as separate JPEG image. Hear is how the algorithm works:

Step 1: The algorithm accepts a scanned image of the questioner



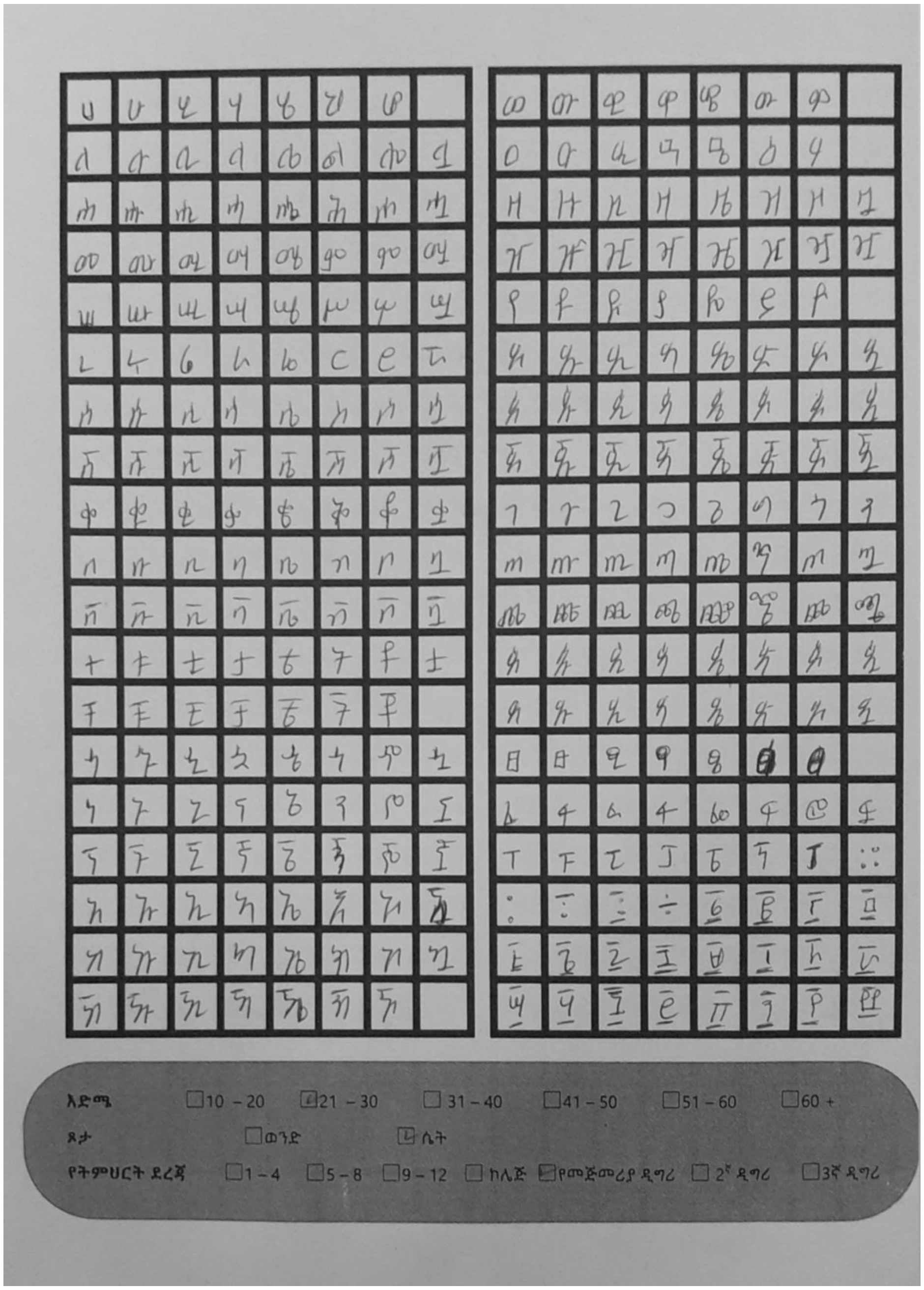
እድሜ □10 – 20 ☒21 – 30 □31 – 40 □41 – 50 □51 – 60 □60 +

ጾታ □ወንድ ☒ሴት

የትምህርት ደረጃ □1 – 4 □5 – 8 □9 – 12 □ከሊጅ ☒የመጀመሪያ ዲግሪ □2ኛ ዲግሪ □3ኛ ዲግሪ

Figure 2: Original scanned image

Step 2: The input image will be resizing to a size of 1200px keeping the proportion of the image. Also, since we don’t need the color the algorithm converts the image in to a gray scale.



እድሜ □10 - 20 ☑21 - 30 □ 31 – 40 □41 – 50 □51 – 60 □60 +

ጾታ □ወንድ ☑ሴት

የትምህርት ደረጃ □1 – 4 □5 – 8 □9 – 12 □ ኮሌጅ ☑የመጀመሪያ ዲግሪ □ 2ኛ ዲግሪ □3ኛ ዲግሪ

Figure 3: Resized gray scale image

Step 3: Next we will try to remove noise from the picture by using gaussian blur algorithm then thresh hold the image buy using OTSU thresh algorithm and invert the image. This means the final image will be a binary image 0 and 1 with black background and white text.

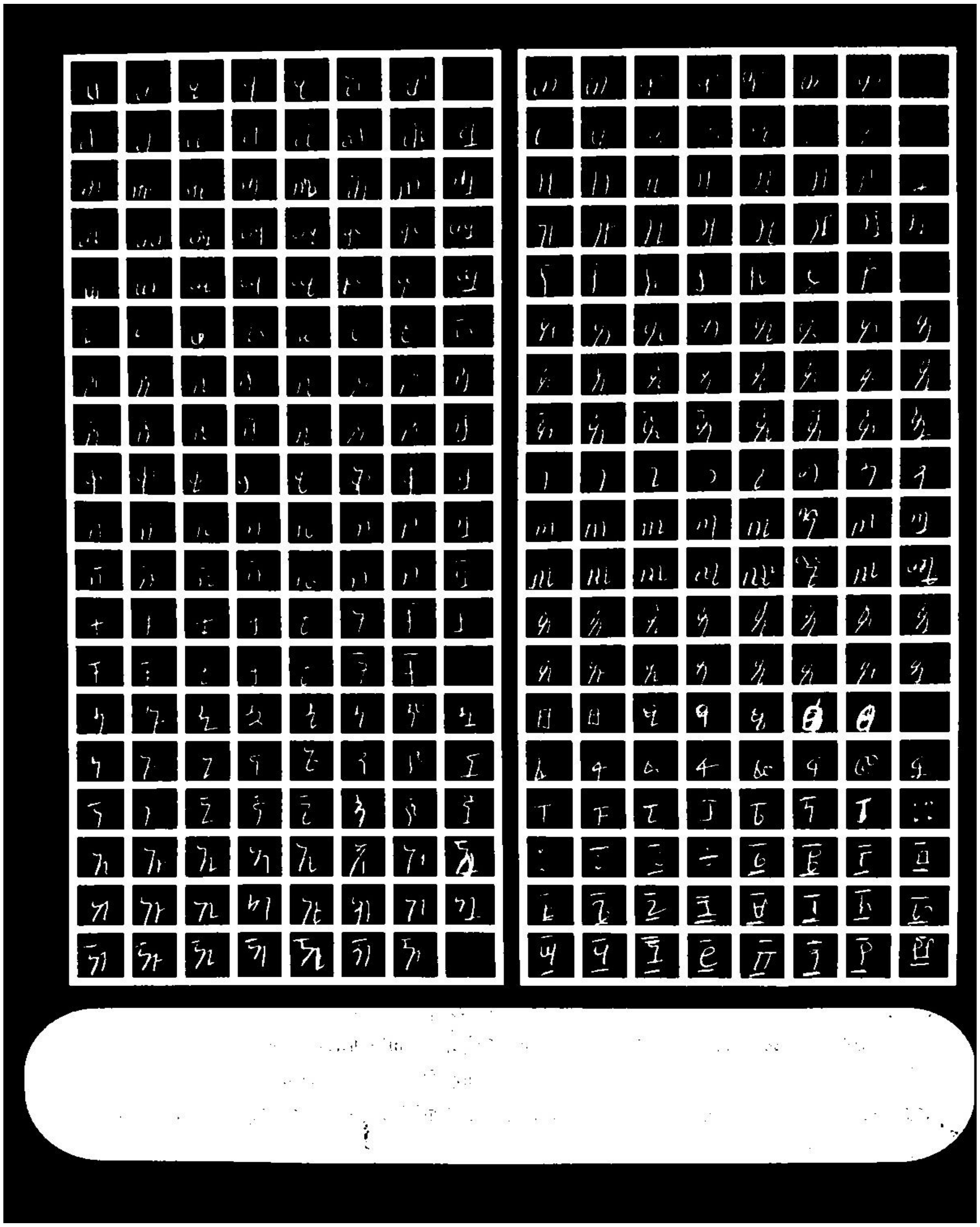


Figure 4: Filtering and OTSU threshold applied to the image

As you can see, we have managed to clear out the image and make it is for farther processing.

Step 4: the algorithm makes some morphological adjustment. This process helps us to clear some damaged edges of the lines during the previous processes.

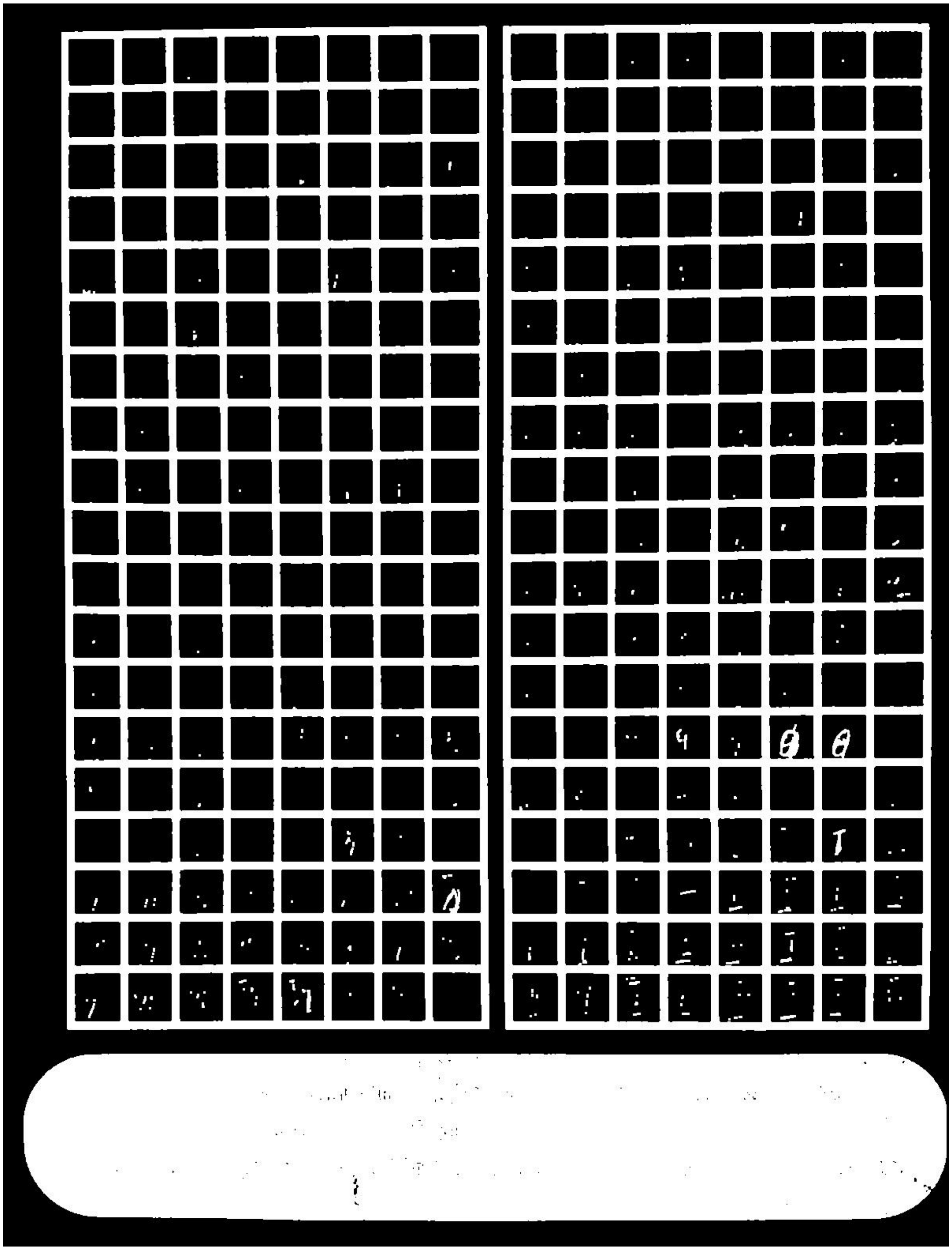


Figure 5: Morphological adjustment is made on the image

Step 5: Now the algorithm tries to detect only horizontal and vertical lines from the image using horizontal and vertical kernels.

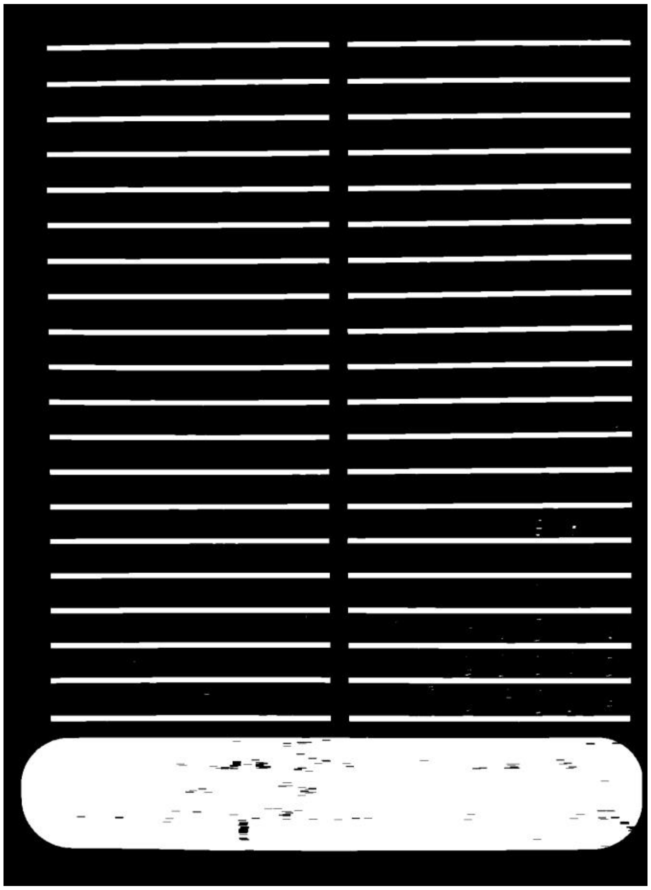


Figure 6: Horizontal lines are extracted from image

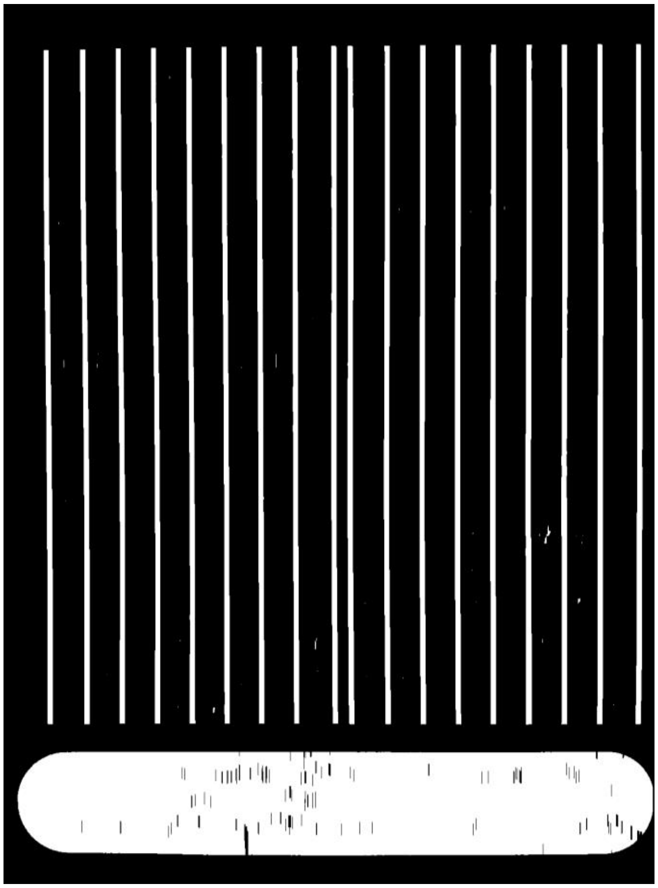


Figure 7: Vertical lines are extracted from image

Step 6: Now the algorithm do bitwise or operation on two images to merge the two images together so that we can get an image which with only the lines of the table.

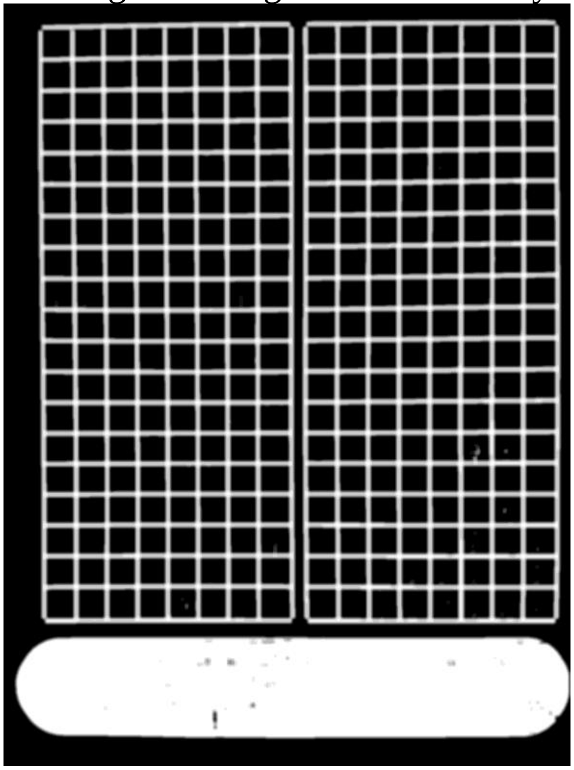


Figure 8: Bitwise or are applied on extracted horizontal lines and extracted vertical lines

Step 7: after we have detected the table, we will use canny edge detection algorithm to get separate boundaries of each box in the table.

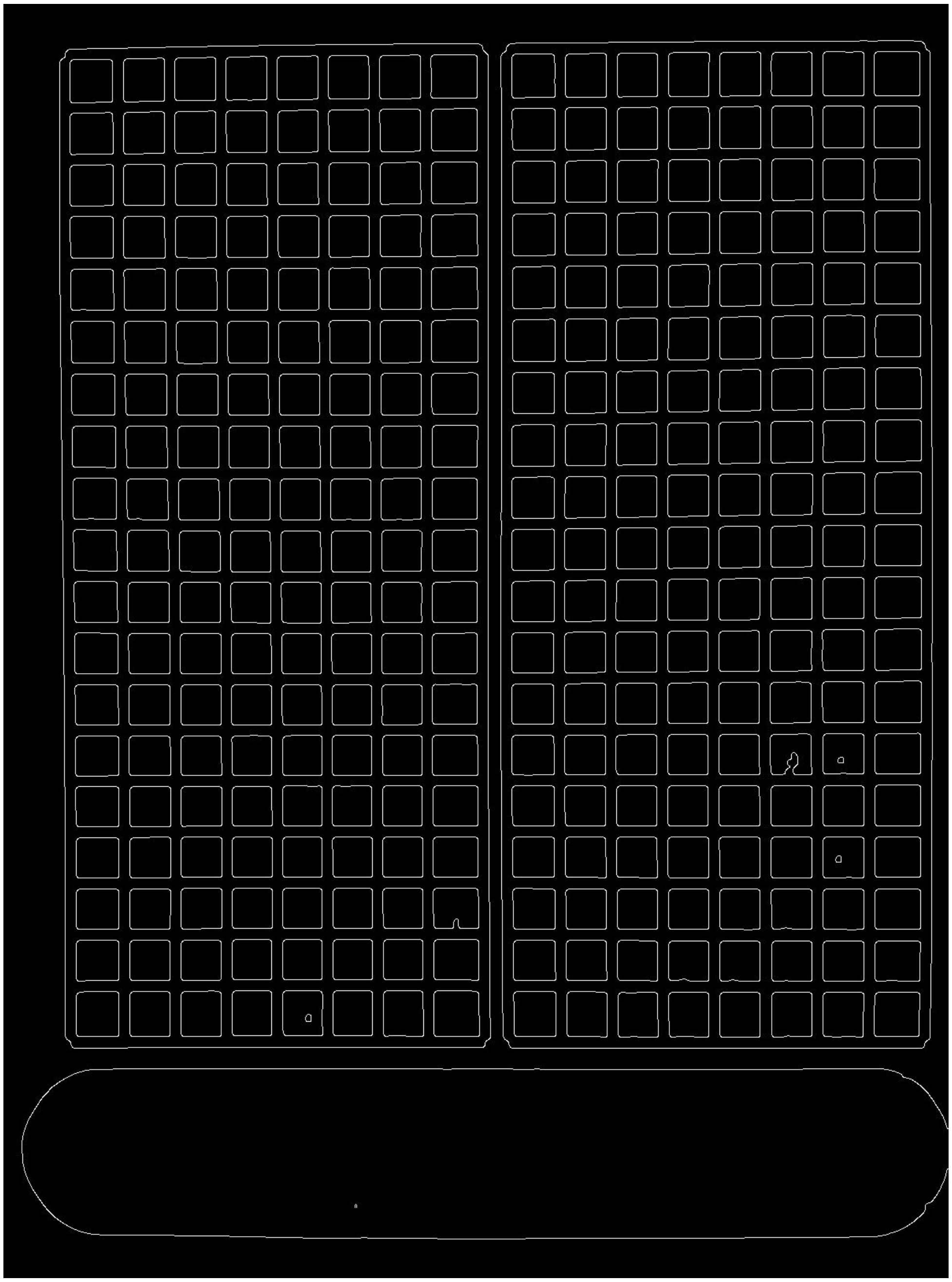


Figure 9: Canny edge detection is applied on the image on figure 7

Step 8: Now, the algorithm has got the location of each box one the image and also the area of the box. In order to prevent form detecting some noise smaller or larger boxes the algorithm only takes boxes with area between the mean area of all boxes and 1000. The algorithm saves the points in array.

Step 9: We have developed a sorting algorithm which uses one kernel which has the width of the image. This kernel scans the image from top to bottom and sort points in same row.

Step 10: Finally, the algorithm has sorted the points. Now it can crop each box containing character and save as a jpg file.

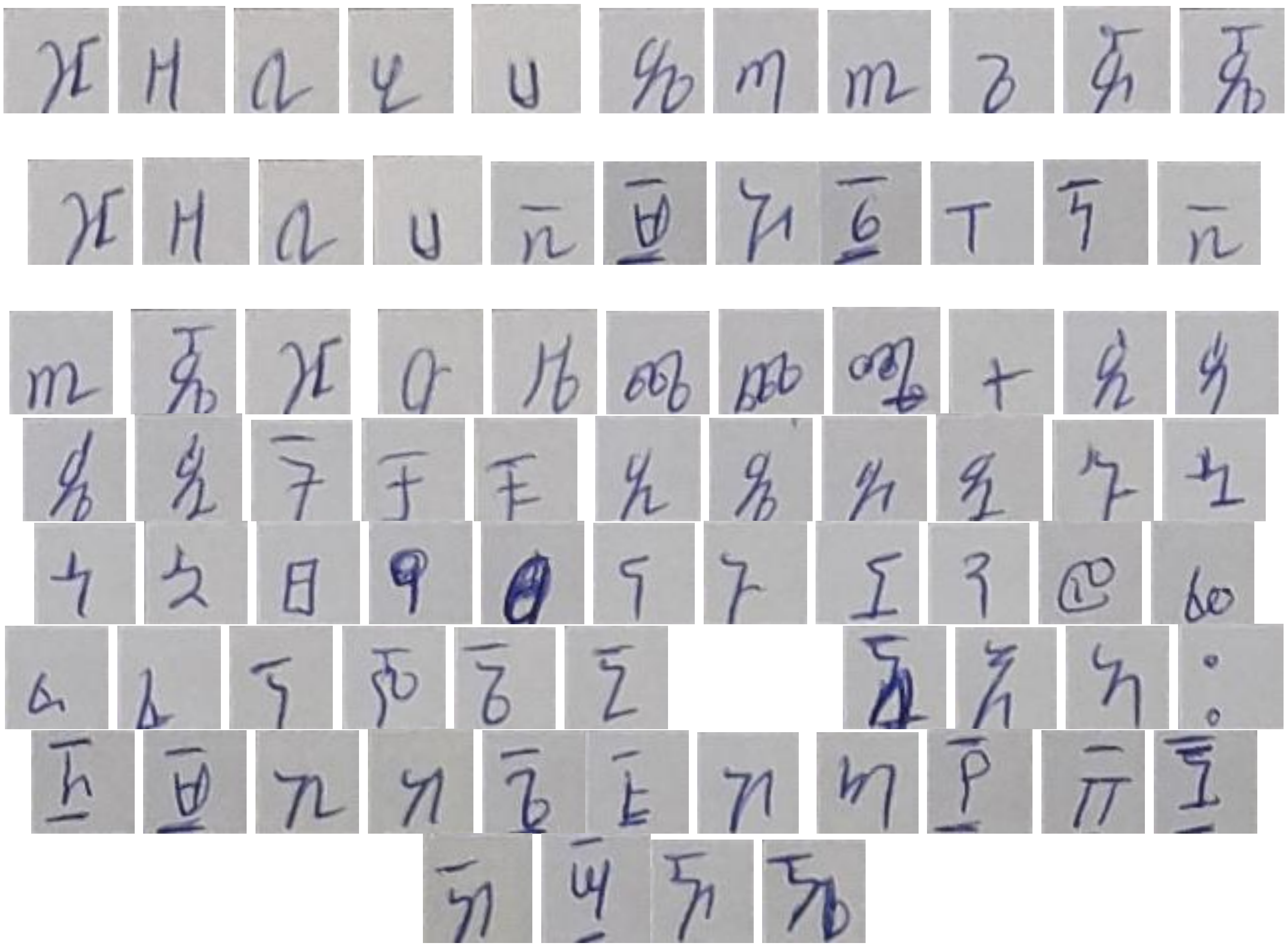


Figure 10: Sample segmented images

In this way the algorithm helps us segment each character in to individual images in automated way.

- **Note**: Sort coordinates algorithm will help as automatically label the images by their position

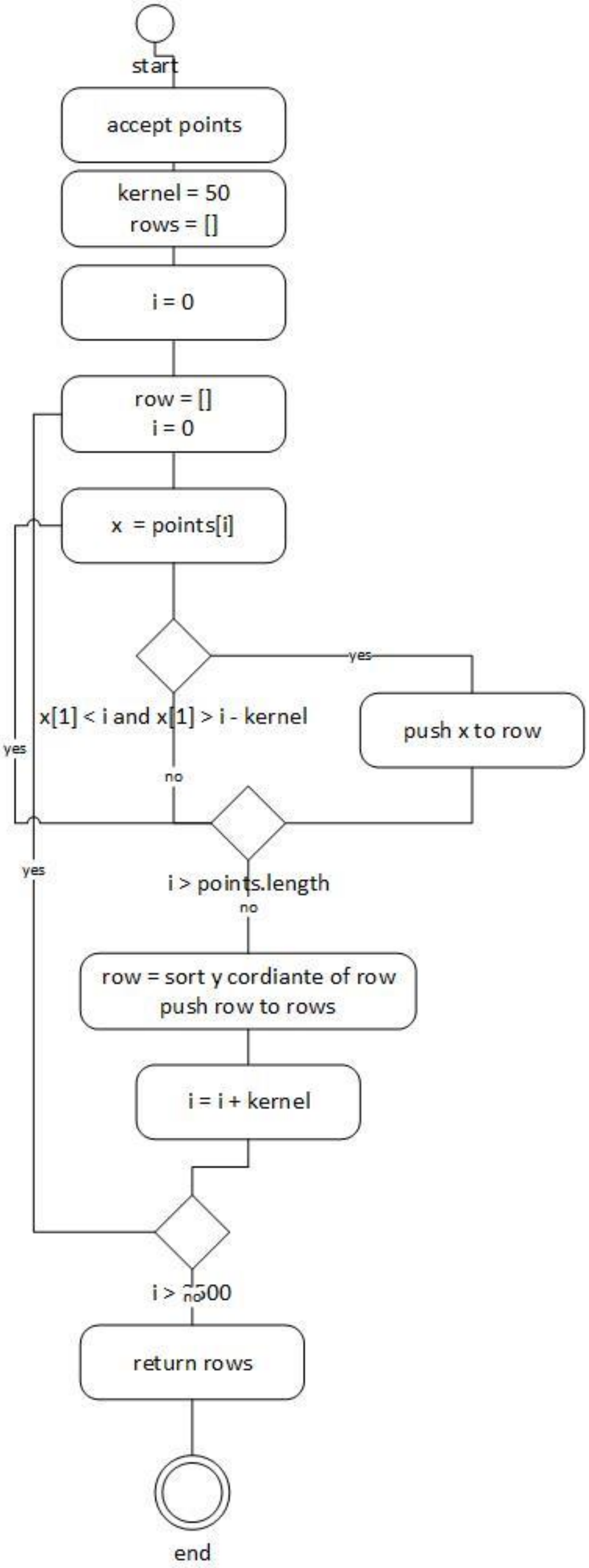


Figure 11: Algorithm for sorting bounding box coordinates

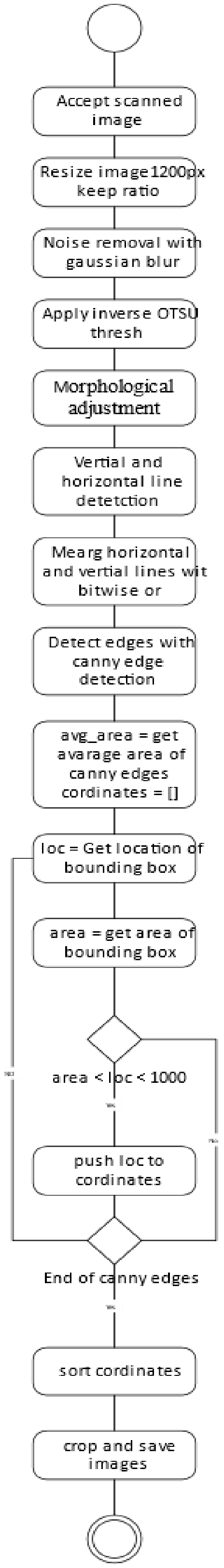

Accept scanned image
Resize image1200px keep ratio
Noise removal with gaussian blur
Apply inverse OTSU thresh
Morphological adjustment
Vertial and horizontal line detetction
Mearg horizontal and vertial lines wit bitwise or
Detect edges with canny edge detection
avg_area = get avarage area of canny edges cordinates = []
loc = Get location of bounding box
area = get area of bounding box
area < loc < 1000
push loc to cordinates
End of canny edges
sort cordinates
crop and save images

Figure 12: Algorithm for segmenting images in the questioner

3.10. Software Requirement specification for data collection app

3.10.4. Functional requirement

- FR 6: Authenticate and authorize data collector
- FR 7: Allow the user to upload questioner
- FR 8: Allow the user to label images
- FR 9: Download a dataset generated form the images

3.10.5. Non-functional requirement

1. Response Time:
   - TR11: The system should have a response time less than 30 seconds for recognition, detection and localization
2. Capacity
   - TR12: The system is expected to handle up to 100 users simultaneously
3. User interface:
   - TR13: The system should have a standard user interface that is easy to use
4. Authorization:
   - TR1 4: only authorized users should label training datasets
   - TR 15: only authorize users should be able to upload scanned images
5. Working hours:
   - TR 16: The system should be available at minimum 95%
6. Device and hardware:
   - TR 17: The system should work on desktop environment
7. Operating systems:
   - TR 18: The system should run on Linux and windows Operating systems
   - TR 19: The system should run on any server operating system but Linux server is recommended.
8. Browser:
   - TR 20: the system should support Google chrome, chromium based operating systems, Mozilla Firefox, opera, safari and edge.

3.10.6. Overview of Proposed system

The proposed system is a web application which automate the data collection and labeling process. The application will have the following futures.

- Upload scanned images
- Segment images
- Allow agents to label images

3.10.7. Business rules

- BR1: Only authorized agents by system admin can upload images
- BR2: Any one can download the dataset
- BR3: System admin create account for agents

3.11. System requirement analysis for data collection application

3.11.1Actor and use case identification

Actor name	**Description**
System admin	The administrator of the system
Agent	A person who is assigned to collect and upload data to the system

Table 1: Actors and their description

3.11.2. Use cases

Use case id	Use case name	Include	Extends
UC01	Login		
UC02	Logout	Login	
UC03	Upload questioner	Login	
UC04	Label images	Login	
UC05	Create agent account	Login	
UC06	Revoke agent access	Login	

Table 2: List of use cases

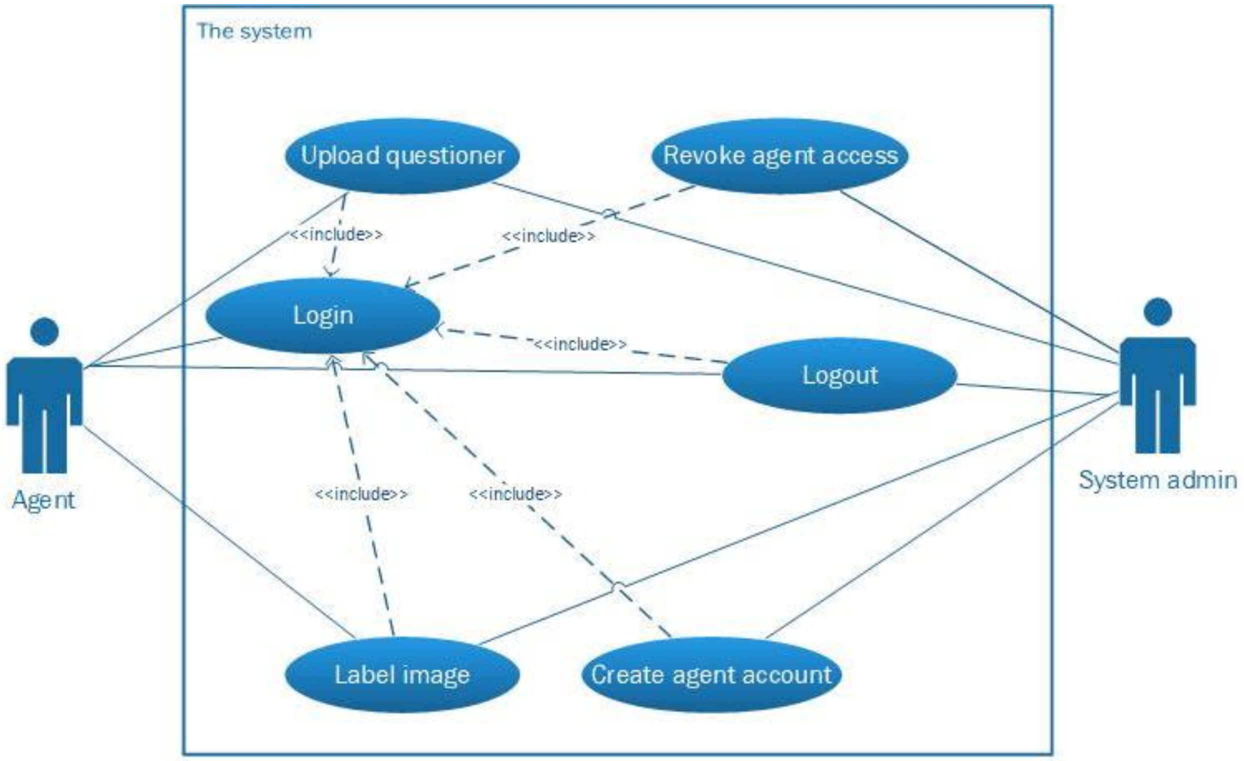


Figure 13: Use case diagram for data collection application

3.11.2.1. Use case description

Use case name	Login

ID	UC01	
Actor	System admin, agent	
Description	The user of the system enter credential to access the system	
Precondition	The user should have proper credentials	
Posttension	The system admin or agent login into the system	
Basic course of action	User action	System response
	1. The user open login page	
		2. The system shows login form
	3. The user enters credentials and press login	
		4. The system checks the credentials 5. Show home page 6. Use case end
An alternative course of action	A: If step 4 failed show error message A1: The system displays and error message A2: Start from step 3	

Table 3: Login use case description

Use case name	Logout	
ID	UC02	
Actor	System admin, agent	
Description	The user logs out of the system	
Precondition	The user should be logged in	
Posttension	The system admin or agent out of the system	
Basic course of action	User action	System response
	1. The user sends logout request	
		2. The system expires authentication key 3. Respond success message 4. Redirect the user to login page
An alternative course of action	A: If step 2 failed show error message	

Table 4: Logout use case description

Use case name	Upload questioner	
ID	UC03	
Actor	System admin, agent	
Description	The user fills the questioner form and upload questioner scanned images	
Precondition	The user should already be logged in to the system	
Posttension	The questioner image will be uploaded and new questioner is created	
Basic course of action	User action	System response
	1. The user navigates to questioner form	
		2. The system shows questioner form
	3. The user fills the questioner form and upload scanned images and submit the form	
		4. The system creates new questioner, segment and save uploaded images. 5. Show success message 6. Use case end
An alternative course of action	A: If step 4 failed show error message	

Table 5: Upload questioner use case description

Use case name	Label images	
ID	UC04	
Actor	System admin, agent	
Description	The user label uploaded images	
Precondition	• The user should already be logged in to the system • A questioner has to already been uploaded	
Posttension	The system labels the image	
Basic course of action	User action	System response
	1. The user selects the questioner and image to label.	

	2. The user navigates to labeling page	
		3. The system shows images and labeling form
	4. The user enters label for each image or verify if it was already been labeled	
		5. The system saves the label of the image 6. Show next image set to label 7. Use case end
An alternative course of action	A: If step 5 failed show error message	

Table 6: Label images use case description

Use case name	Create agent account	
ID	UC05	
Actor	System admin	
Description	The system admin creates agent account	
Precondition	The system admin should be logged in as system admin role	
Posttension	New agent is created	
Basic course of action	User action	System response
	1. Navigate to create agent form	
		2. The system shows create agent form
	3. Fill the create agent form login	
		4. The system creates new agent 5. Show list of agent page 6. Use case end
An alternative course of action	A: If step 4 failed show error message	

Table 7: Create agent account use case description

Use case name	Revoke agent access	
ID	UC06	
Actor	System admin	
Description	The system admin revoke access from the agent	
Precondition	The system admin should be logged in as system admin role	
Posttension	The agent can no longer access the system	
Basic course of action	User action	System response
	1. Navigate to agents list and press revoke button	
		2. The system shows confirmation message
	3. Confirm revoking access of the agent	
		4. The system revokes the access of the user 5. Show list of agent page 6. Use case end
An alternative course of action	A: If step 4 failed show error message A1: If the user cancel confirmation on step 3 the revoke process will be canceled and go back to step 1	

Table 8: Revoke agent access use case description

3.11.3. Sequence Diagram

Login Sequence diagram

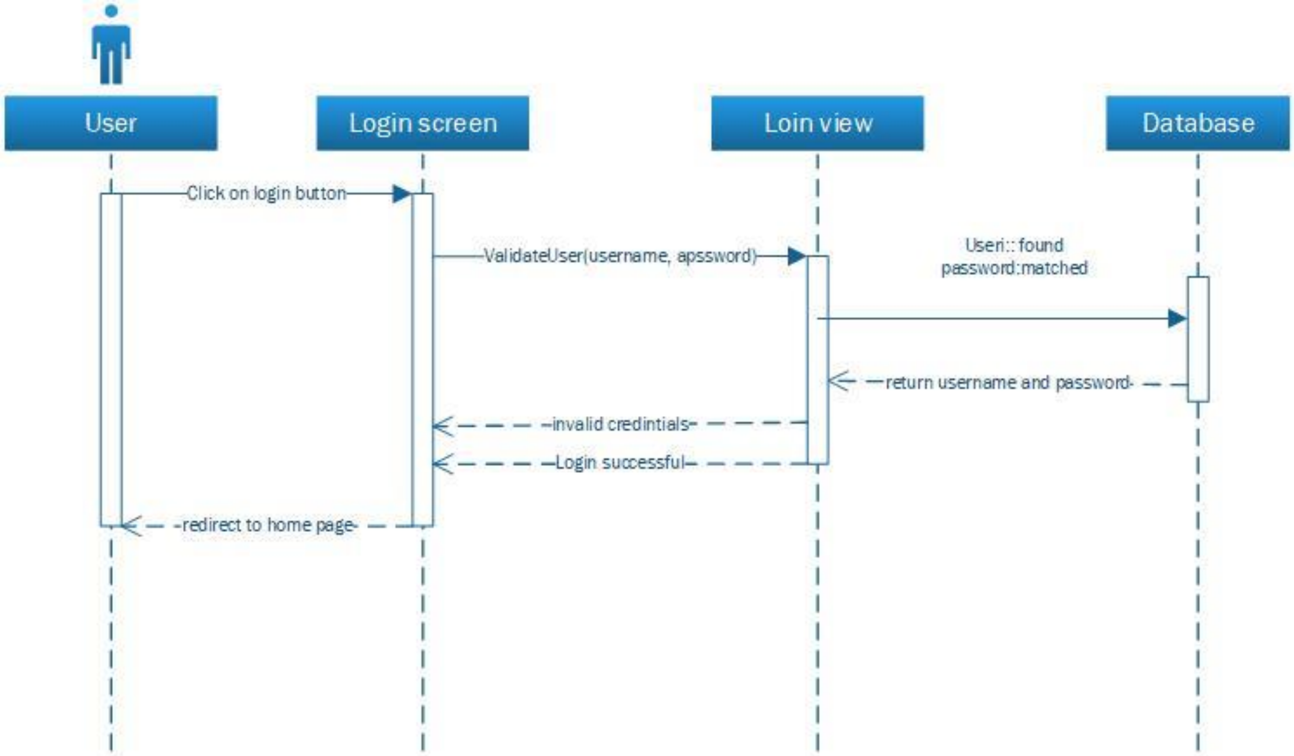


Figure 14: Sequence diagram for login

Logout Sequence diagram

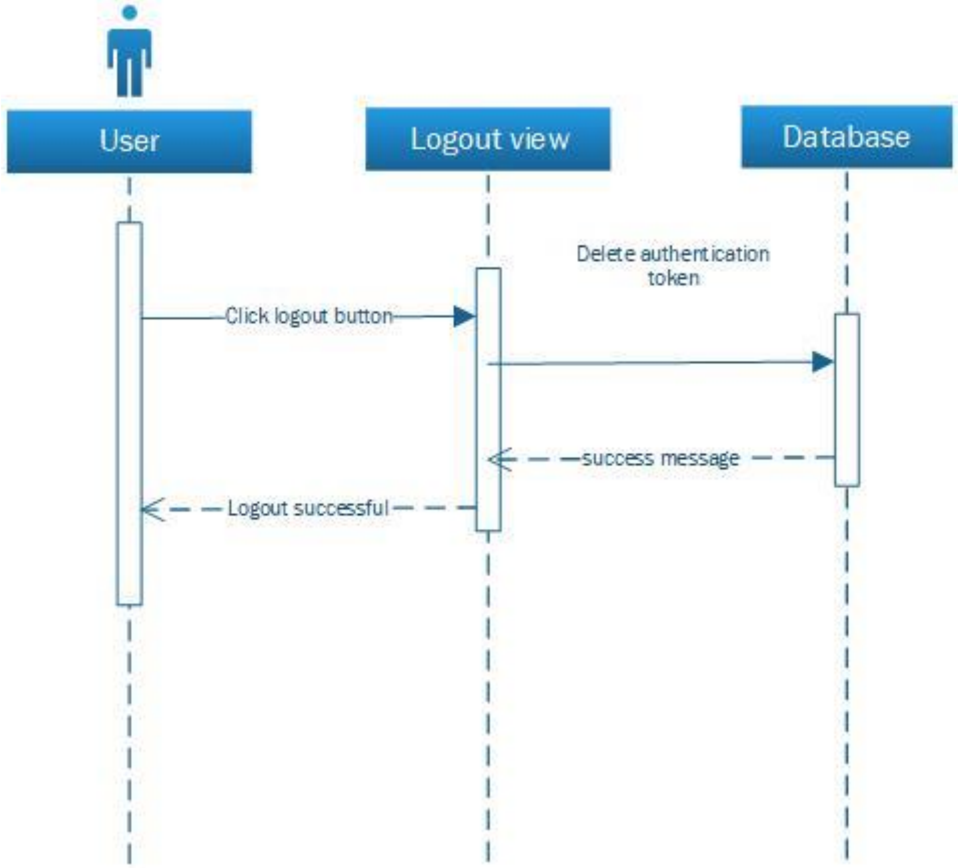


Figure 15: Logout sequence diagram

Create user sequence diagram

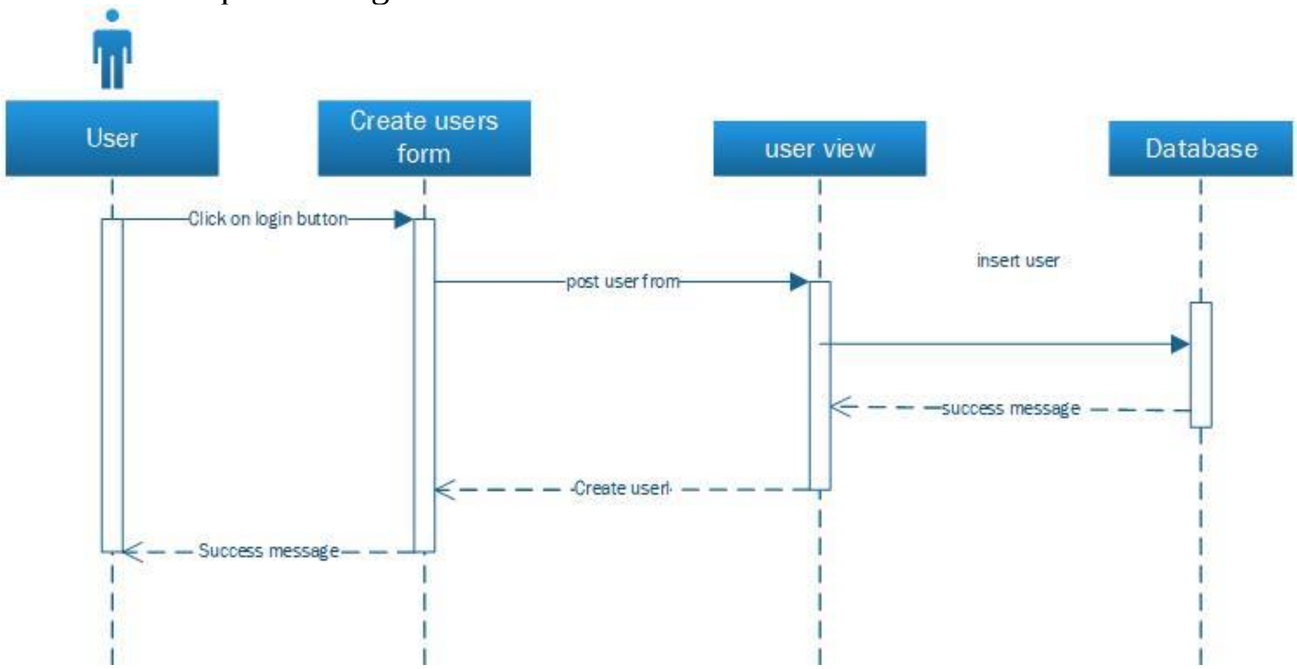


Figure 16: Create user sequence diagram

Label image sequence diagram

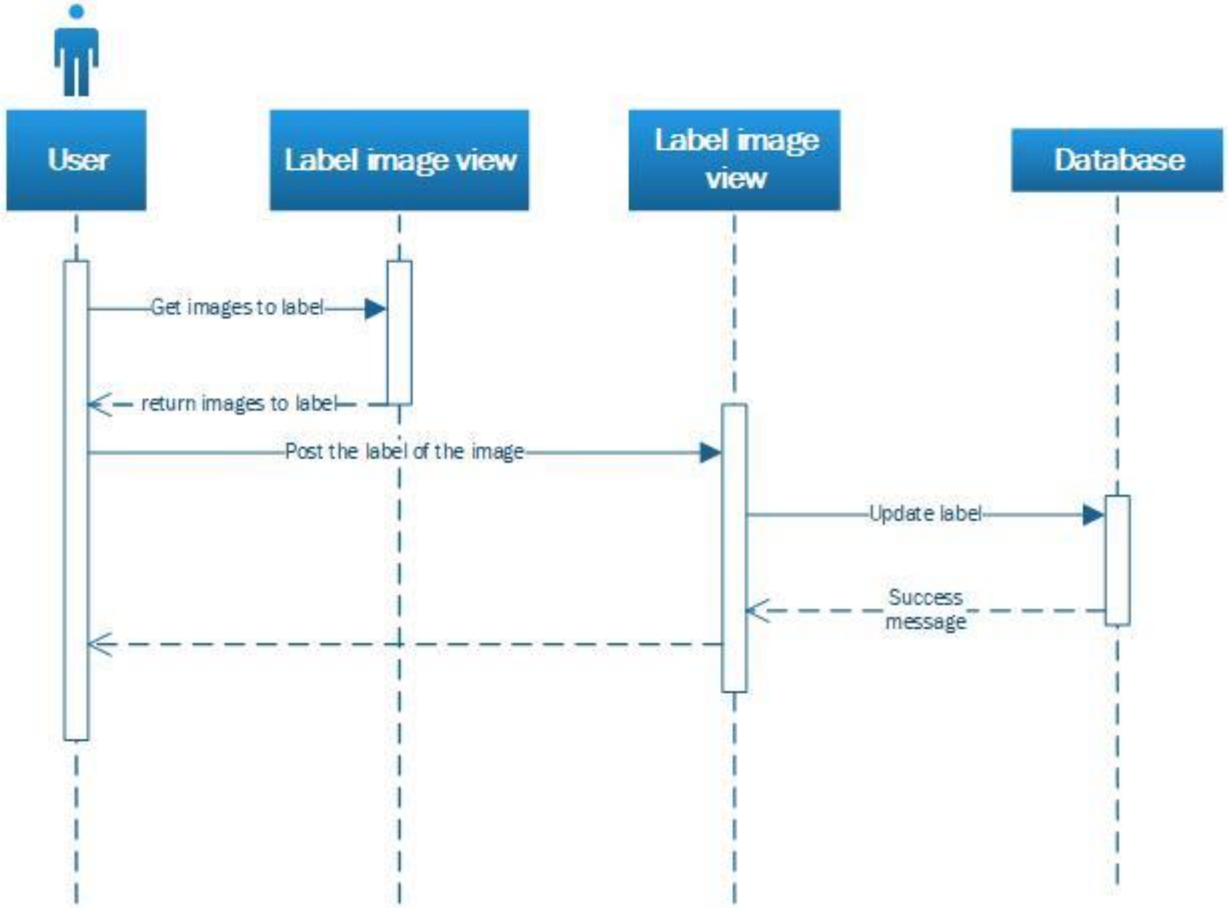


Figure 17: Label images sequence diagram

Upload questioner sequence diagram

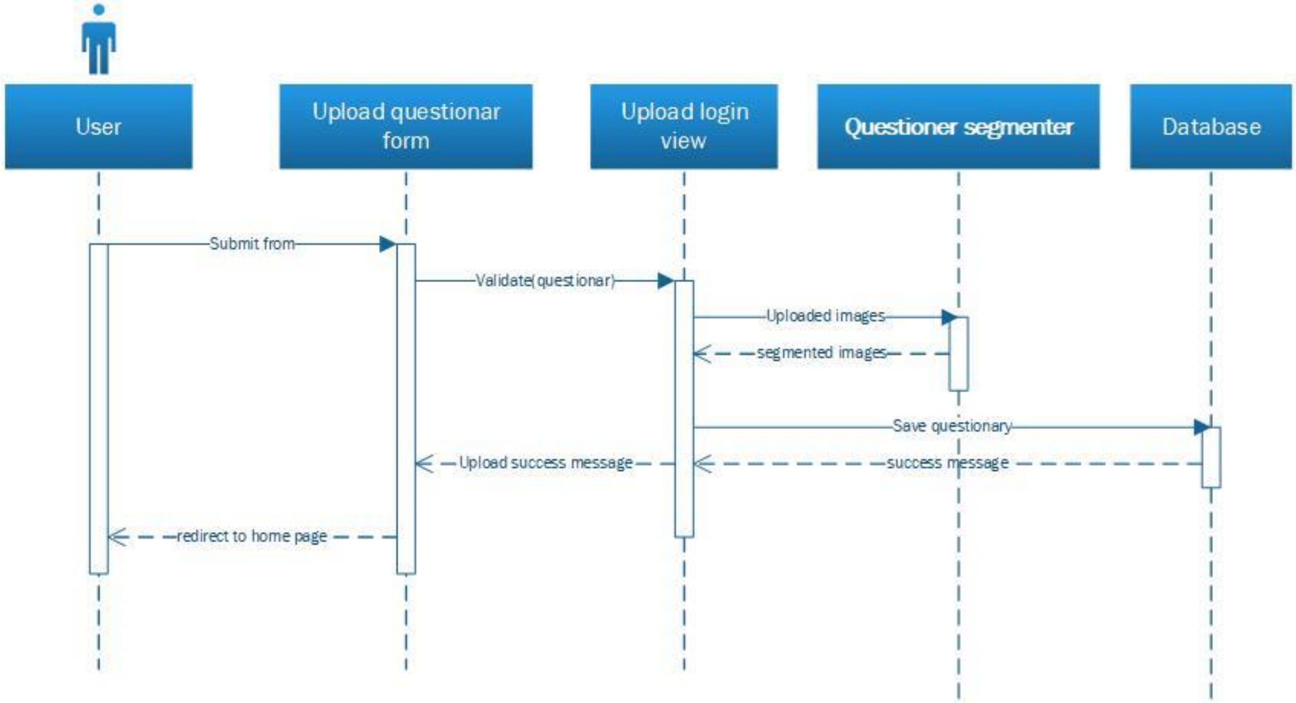


Figure 18: Upload questioner sequence diagram

3.11.4. Activity Diagram

The following activity diagram shows how agents label images.

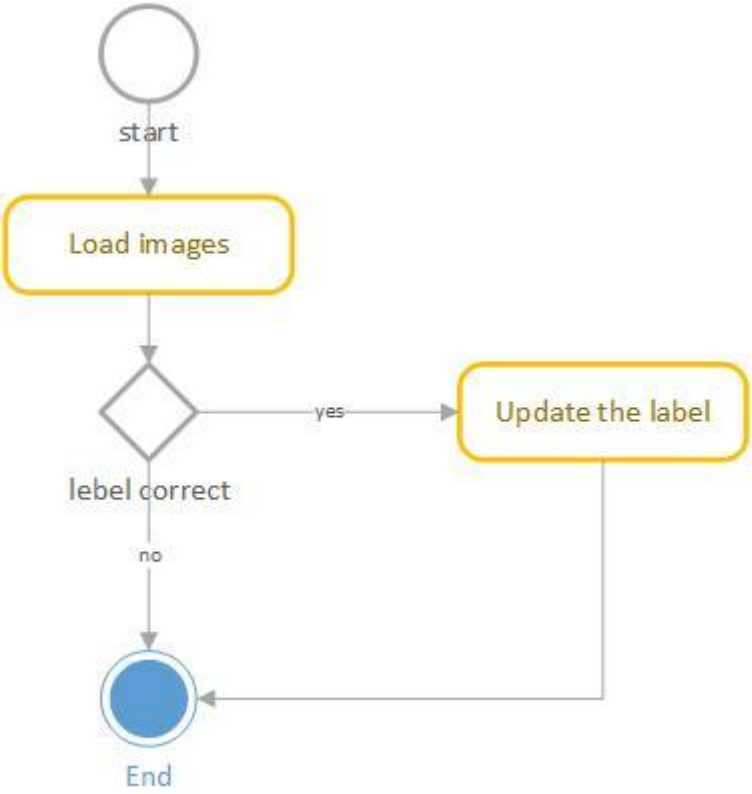


Figure 19: Figure 16: Label image sequence diagram

3.11.5. Analysis Class Diagram

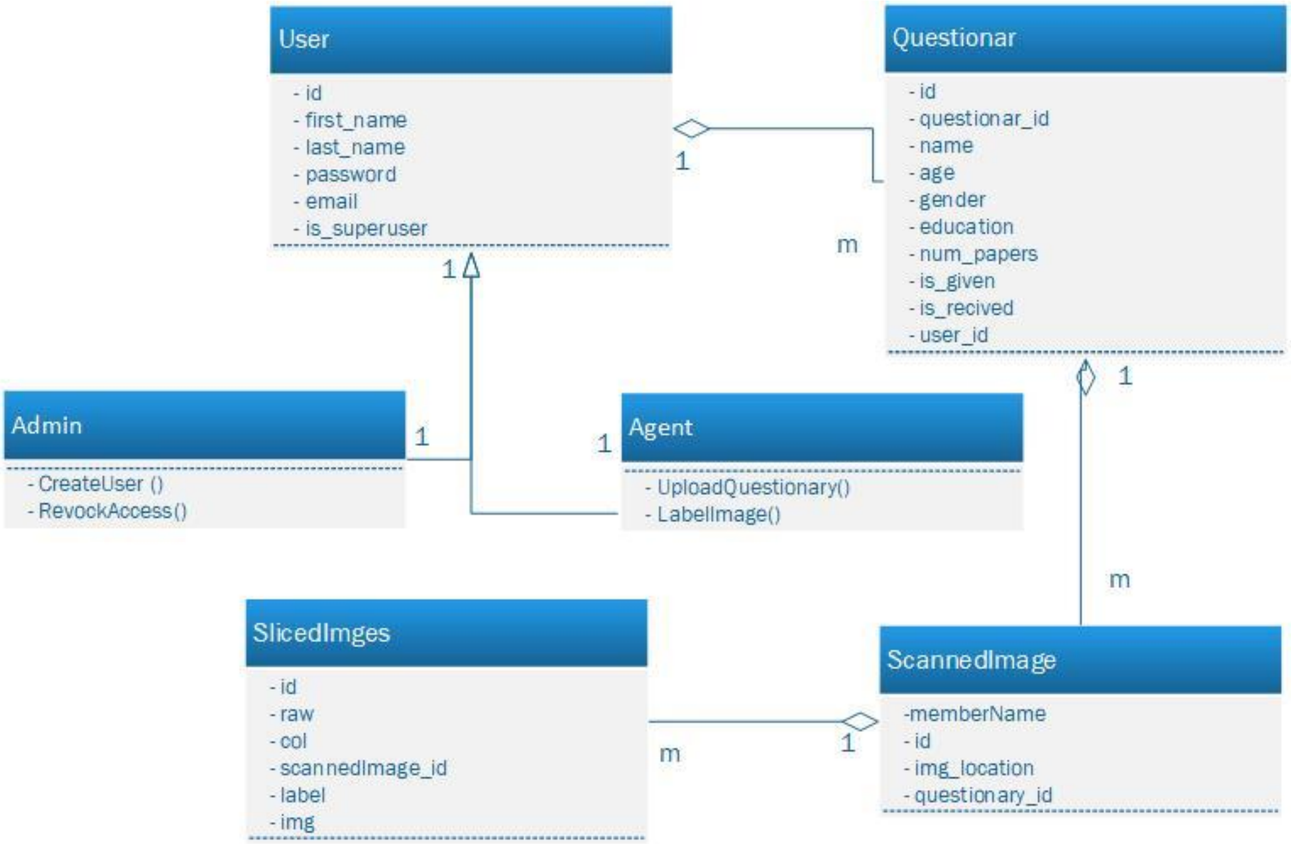


Figure 20: Analysis class diagram

3.12. System Design

3.12.1. Design Class Diagram

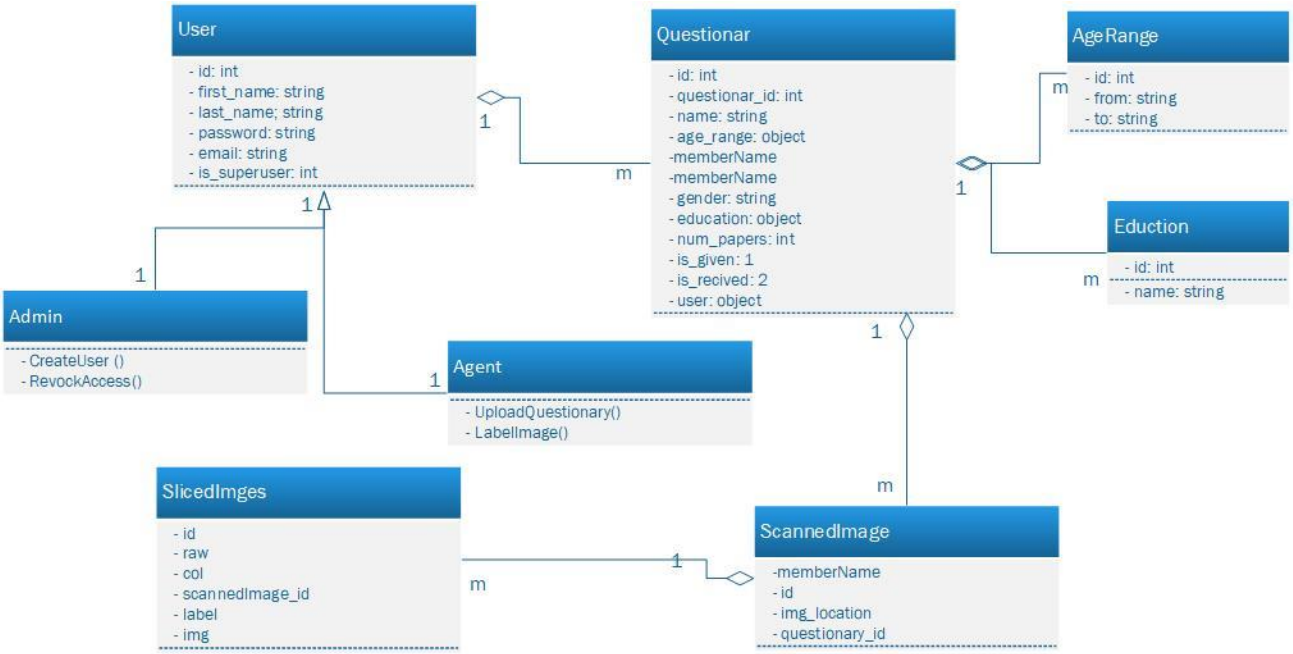


Figure 21: Design class diagram

3.12.2. Physical Data Model

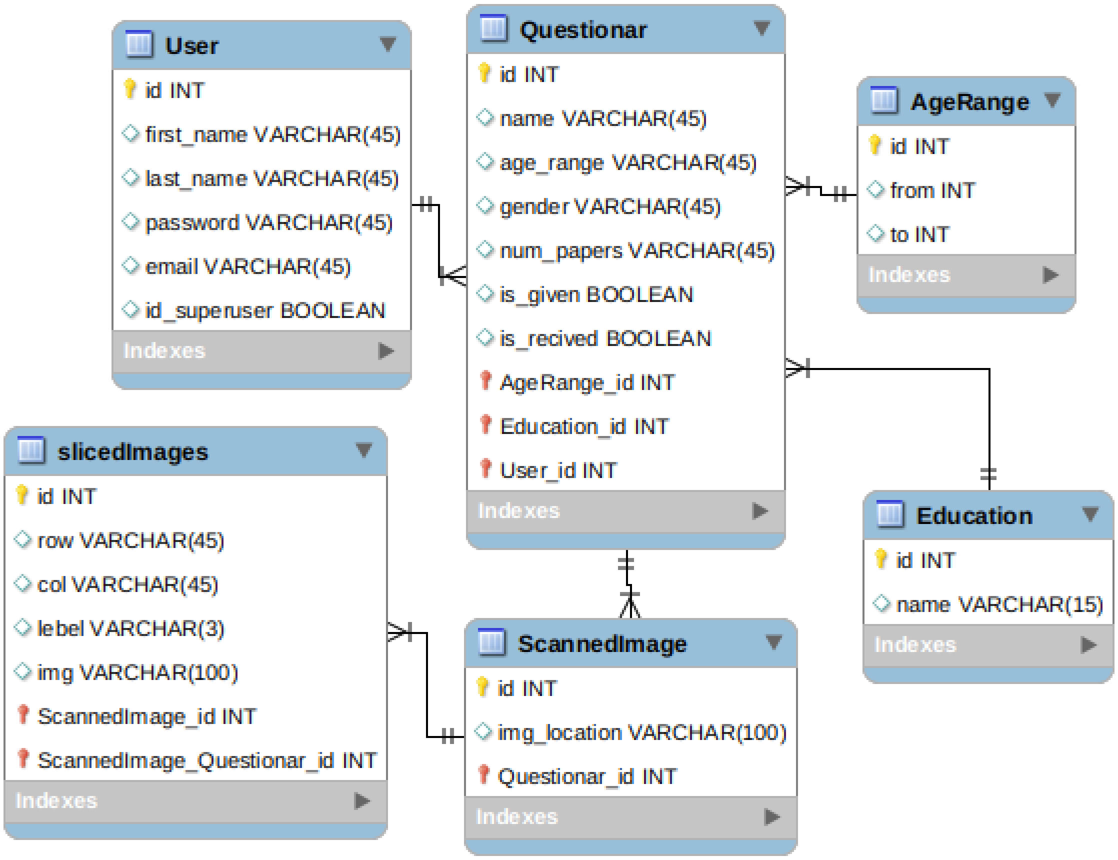


Figure 22: Physical data model

3.12.3. User Interface Design

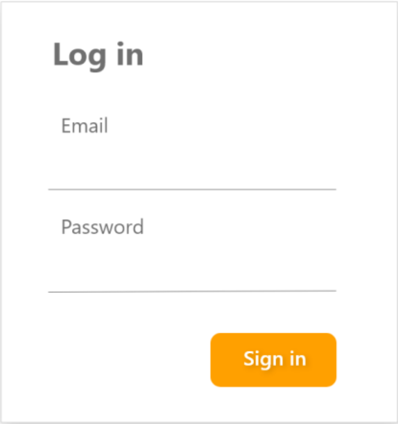


Figure 23: Login interface

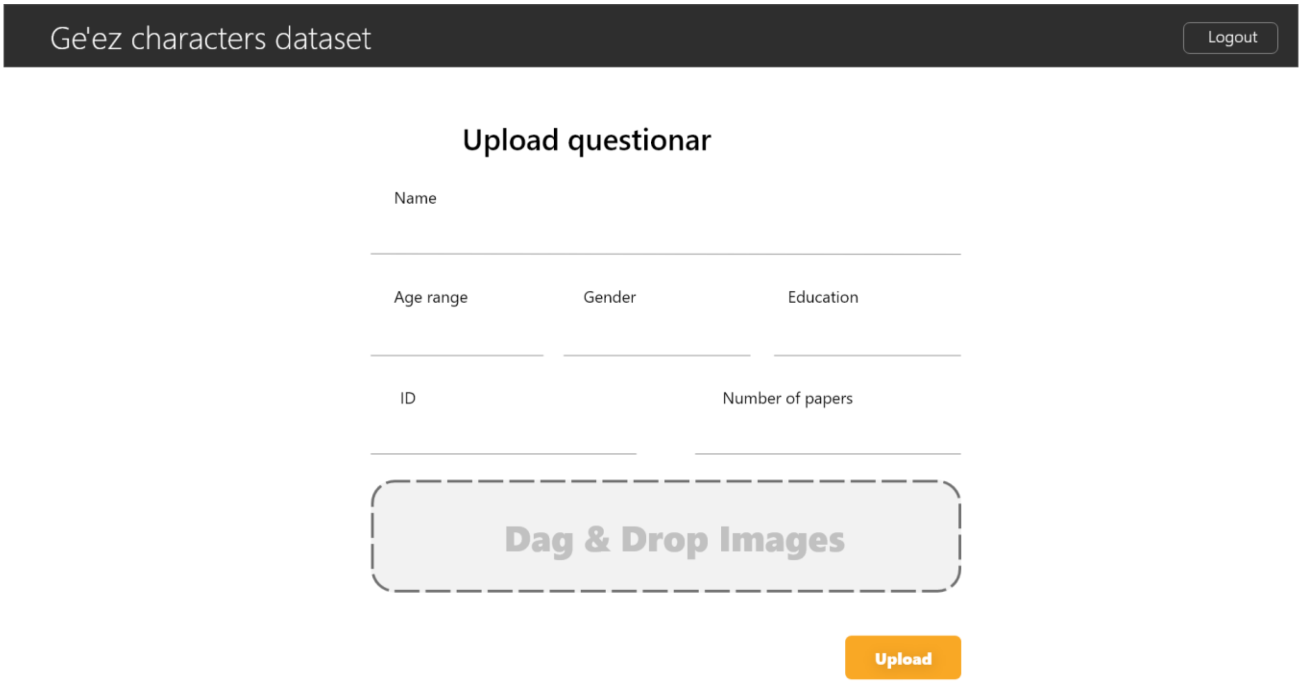


Figure 24: Upload questioner form

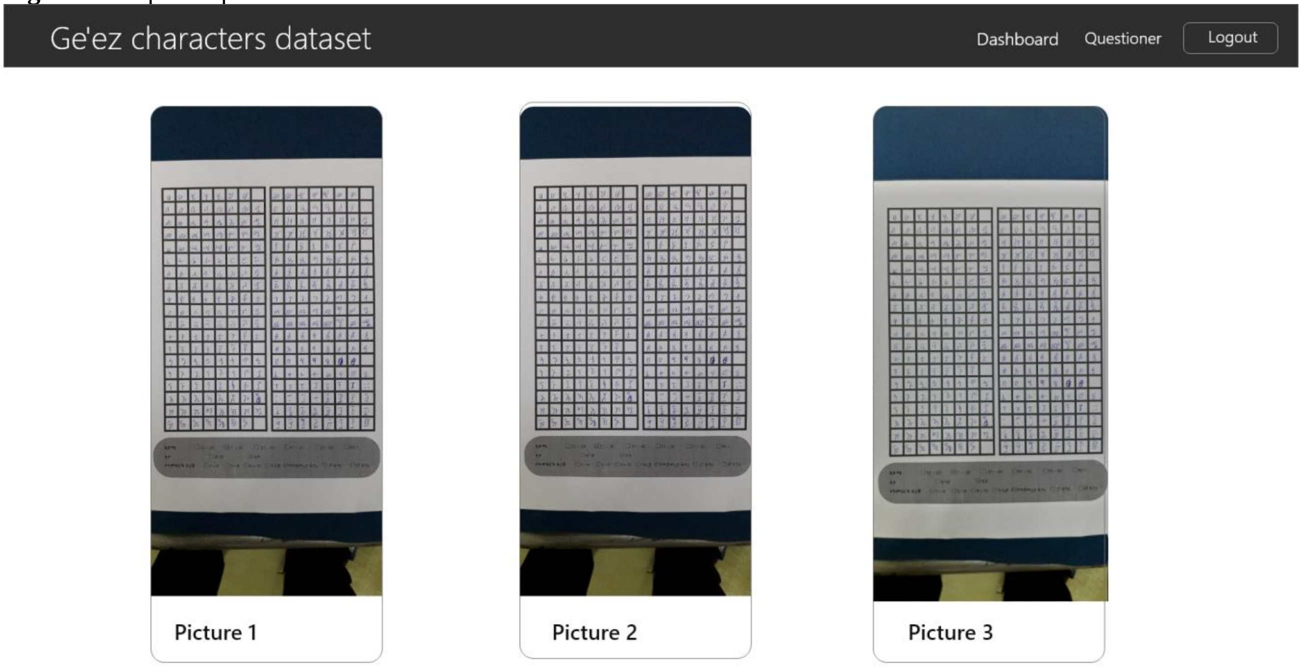
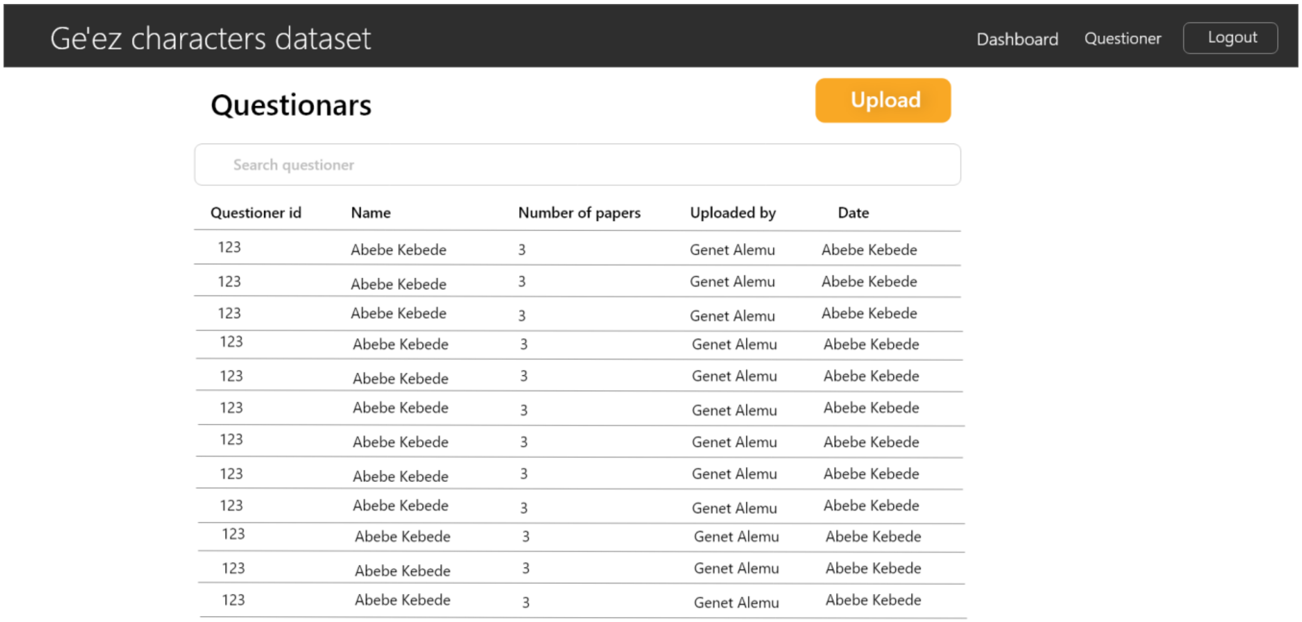


Figure 25: Questioner images list



Ge'ez characters dataset

Questionars

Questioner id	Name	Number of papers	Uploaded by	Date
123	Abebe Kebede	3	Genet Alemu	Abebe Kebede
123	Abebe Kebede	3	Genet Alemu	Abebe Kebede
123	Abebe Kebede	3	Genet Alemu	Abebe Kebede
123	Abebe Kebede	3	Genet Alemu	Abebe Kebede
123	Abebe Kebede	3	Genet Alemu	Abebe Kebede
123	Abebe Kebede	3	Genet Alemu	Abebe Kebede
123	Abebe Kebede	3	Genet Alemu	Abebe Kebede
123	Abebe Kebede	3	Genet Alemu	Abebe Kebede
123	Abebe Kebede	3	Genet Alemu	Abebe Kebede
123	Abebe Kebede	3	Genet Alemu	Abebe Kebede
123	Abebe Kebede	3	Genet Alemu	Abebe Kebede
123	Abebe Kebede	3	Genet Alemu	Abebe Kebede

Figure 26: Questionnaires list interface design

3.12.4. Deployment Design

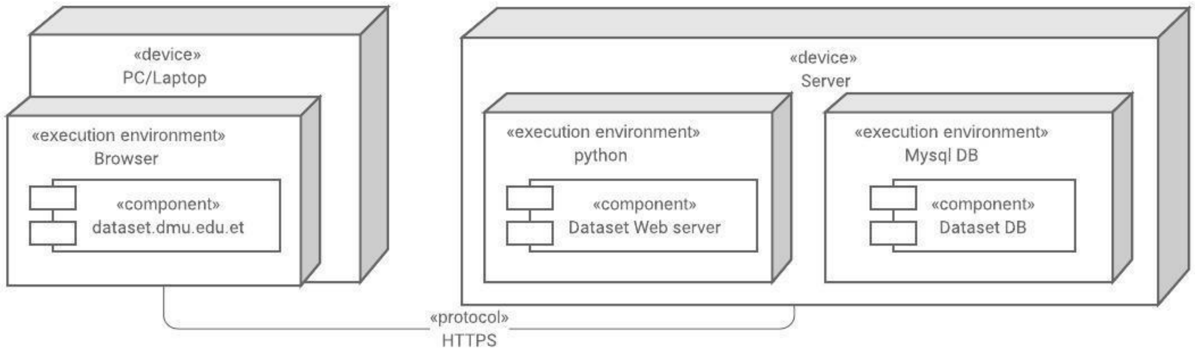


Figure 27: Display design diagram

4. Implementation

4.1. Overview of the programming language used

- **Python** is used for image processing, model training and other automation tasks
- **Drat** language is used for the android application.

4.2. Algorithms Used

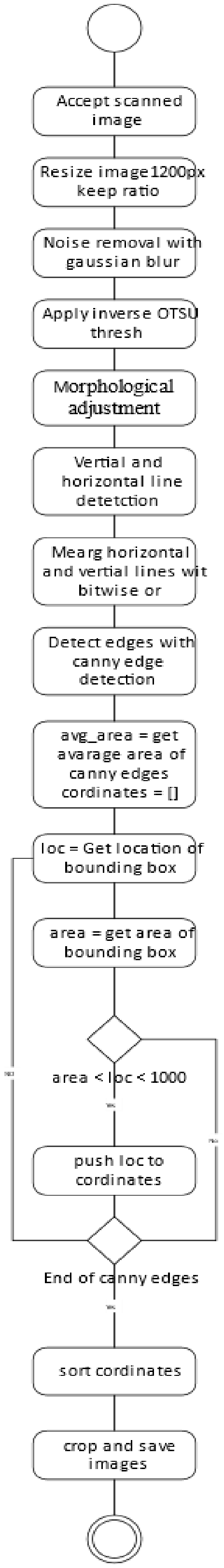

Accept scanned image
Resize image1200px keep ratio
Noise removal with gaussian blur
Apply inverse OTSU thresh
Morphological adjustment
Vertial and horizontal line detetction
Mearg horizontal and vertial lines wit bitwise or
Detect edges with canny edge detection
avg_area = get avarage area of canny edges cordinates = []
loc = Get location of bounding box
area = get area of bounding box
area < loc < 1000
push loc to cordinates
End of canny edges
sort cordinates
crop and save images

Figure 28 Slicing algorithms flow chart

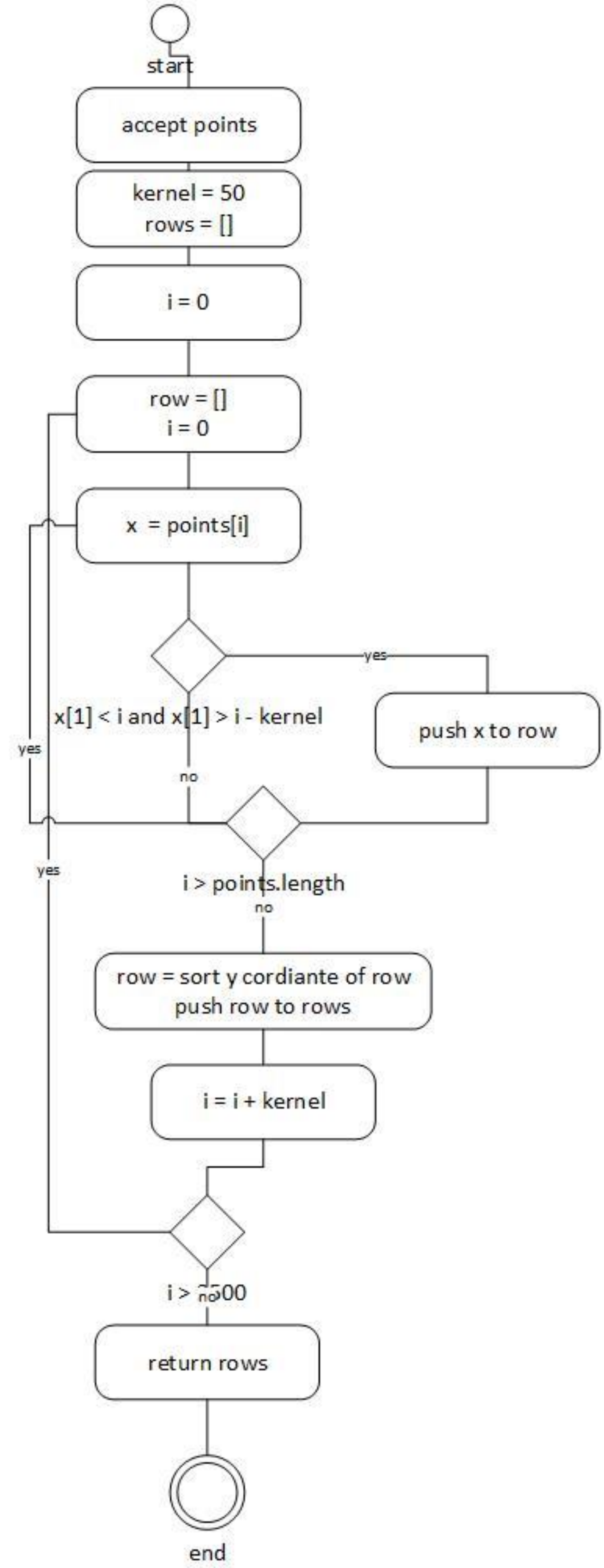


Figure 29: Sort coordinates algorithms flow chart

4.3. Sample Codes

4.3.1. Image Slicer

The main purpose of this algorithm is to extract a character box form the scanned data collection questioner.

In [1]:

```
import math
def distance(x1, y1, x2, y2):
    return math.sqrt(pow(x2-x1, 2) + pow(y2-y1,2))
```

Sorting algorithm is used to sort coordinate points horizontally and vertically. The contour algorithm detects the bounding boxes of every object on the image. After objects are detected they will be filtered by there size and other parameters. Then this algorithm sort the (x, y) coordinates of this objects form left to right and top to bottom. It uses a kernel to stride from top to bottom.

In [2]:

```
def _sort(points):
    kernel = 1250 / 19
    rows = []
    j = round(kernel/2)
    i = j
    while True:
        row = [x for x in points if x[1] < i and x[1] > i - kernel]
        row = sorted(row, key=lambda ctr: ctr[0],  reverse=True)
        if len(row) > 0:
            rows.append(row[::-1])
        j += kernel
        i = round(j)
        if j > 1250 + kernel:
            break


    return rows
```

In [3]:

```
import cv2 as cv
import numpy as np
import sys
from matplotlib import pyplot as plt
import imutils
```

Let us import the scanned image.

In [4]:

```
img_dir ="../dataset/raw-files/MX-M464N_20200211_204539_014.tif"
```

In [5]:

```
img = {}
img_size = 1200
kernel_width_ratio = 16
kernel_height_ratio = 28
```

In [6]:

```
img["orginal"] = cv.imread(img_dir)
print(img["orginal"][0][0])
img["orginal"] = cv.resize(img['orginal'],(img_size, int((img_size *
img["orginal"].shape[0])/img["orginal"].shape[1])))
img["orginal"] = cv.cvtColor(img["orginal"], cv.COLOR_BGR2GRAY)
[255 255 255]
```

This is how the scanned image looks like after coverting it to grayscale image and resizing it to 1200px width

In [7]:

```
plt.imshow(img["orginal"], 'gray')
```

Out[7]:

```
<matplotlib.image.AxesImage at 0x7fbe0f025320>
```

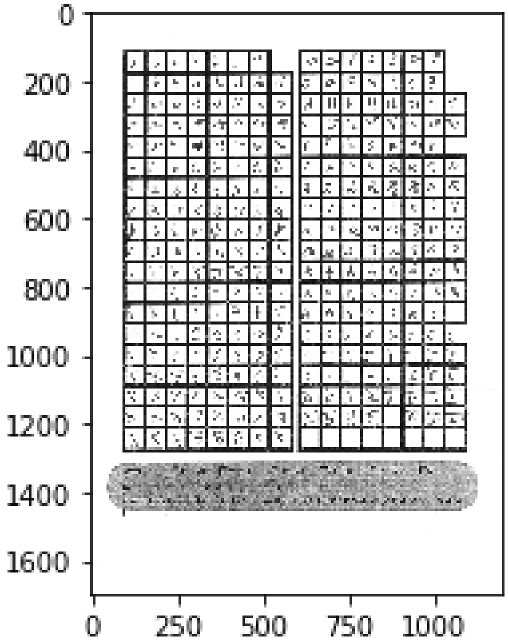


Figure 30

To simplify the image, we changed the gray scale image to binary image by (128, 255) threshold.

In [8]:

```
img["filtered"] = cv.GaussianBlur(img["orginal"], (3, 3),cv.BORDER_CONSTANT)
_, img["trash hold"] = cv.threshold(img["filtered"], 128, 255,
cv.THRESH_BINARY_INV | cv.THRESH_OTSU)
```

In [9]:

```
plt.imshow(img["trash hold"], 'gray')
```

Out[9]:

```
<matplotlib.image.AxesImage at 0x7fbe0eaff5c0>
```

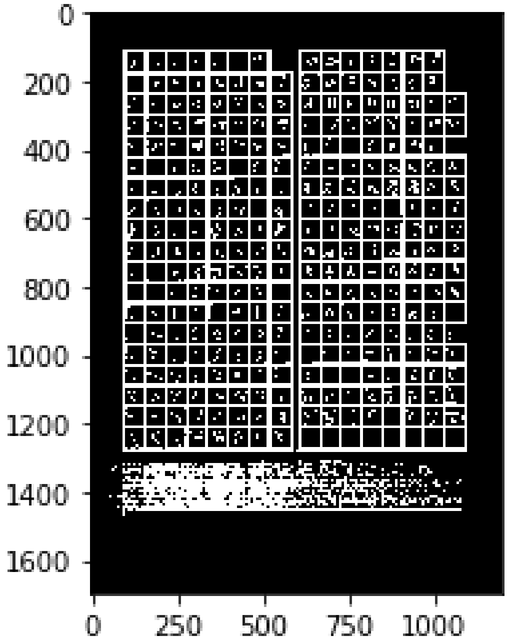


Figure 31

By using morphological transformation we will try to smoothen the edges of objects on the image.

In [10]:

```
opening_kernel = np.ones((3,3),dtype=np.uint8)
img["opened"] = cv.morphologyEx(img["trash hold"], cv.MORPH_OPEN,opening_kernel)
```

In [11]:

```
plt.imshow(img["opened"], 'gray')
```

Out[11]:

```
<matplotlib.image.AxesImage at 0x7fbe0ea69ac8>
```

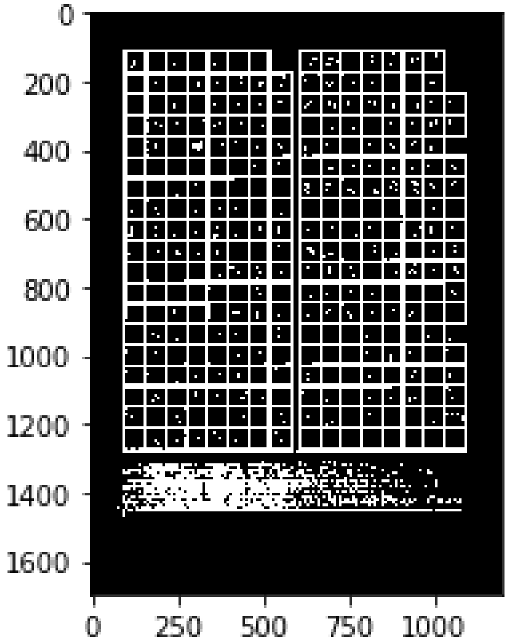


Figure 32

Now we want to extract only the lines of the table. In order to do that we will run horizontal and vertical kernels, which are used to detect the horizontal and vertical line. After that by doing bitwise or operation on the image we can get the table line as showed on the picture below.

In [12]:

```
horizontal_kernel_size = (int(img["orginal"].shape[1] / kernel_width_ratio) //5
, 1)
vertical_kernel_size = (1, int(img["orginal"].shape[0] / kernel_height_ratio)
//4)

horizontal_kernel = cv.getStructuringElement(cv.MORPH_RECT,
horizontal_kernel_size)
vertical_kernel = cv.getStructuringElement(cv.MORPH_RECT, vertical_kernel_size)
```

In [13]:

```
img["horizontal_erod"]  = cv.erode(img["trash hold"], horizontal_kernel,
iterations=1)
img["vertical erod"] = cv.erode(img["trash hold"], vertical_kernel,
iterations=1)

img["extracted image"] = cv.bitwise_or(img["horizontal_erod"], img["vertical
erod"])
```

In [14]:

```
plt.imshow(img["extracted image"], 'gray')
```

Out[14]:

```
<matplotlib.image.AxesImage at 0x7fbe09fe10f0>
```

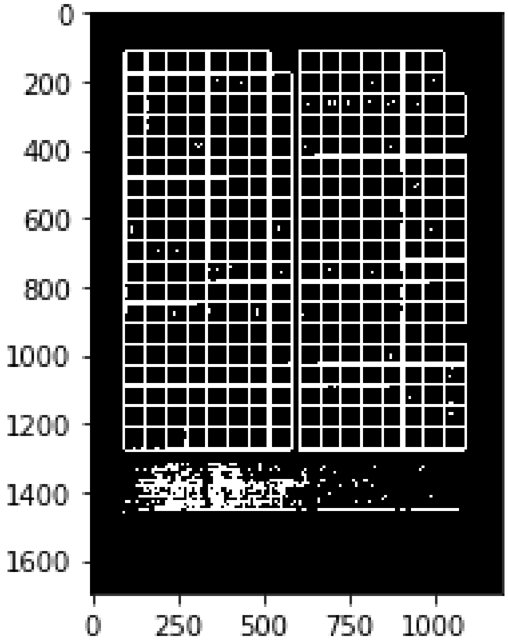


Figure 33

In [15]:

```
img["extracted image"] = cv.GaussianBlur(img["extracted image"], (9, 9),
cv.BORDER_DEFAULT)
img["Extracted image_e"] = cv.morphologyEx(img["extracted image"],cv.MORPH_OPEN,
np.ones((3,3), np.uint8), iterations=2)
_, img["ex_thresh"] = cv.threshold(img["Extracted image_e"], 50, 255,
cv.THRESH_BINARY)
img["edge"] = cv.Canny(img["ex_thresh"], 100, 200)
```

In [16]:

```
plt.imshow(img["edge"], 'gray')
```

Out[16]:

```
<matplotlib.image.AxesImage at 0x7fbe09fc36a0>
```

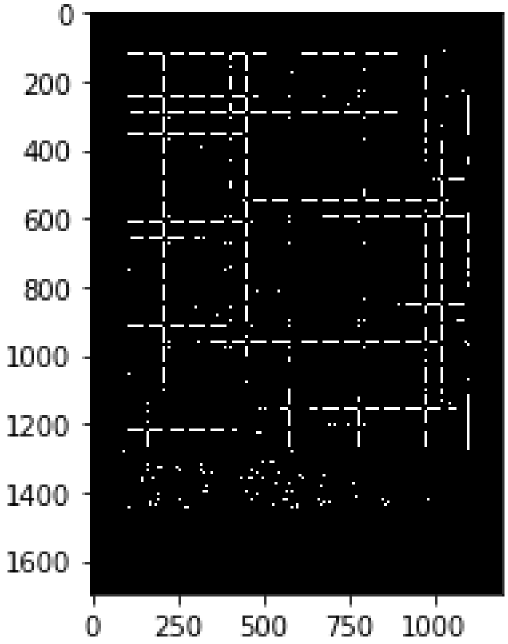


Figure 34

Canny edge detection will give us isolated boxes. By running find Contours algorithm we can detect the coordinates of each isolated boxes.

In [17]:

```
contours, _ = cv.findContours(img["edge"],cv.RETR_TREE,cv.CHAIN_APPROX_SIMPLE)
max_contor = max(contours, key = cv.contourArea)
x, y, w, h = cv.boundingRect(max_contor)
img["edge_croped"] = img["edge"][y: y+h, x: x+ w*2]
img["orginal_croped"] = img["orginal"][y: y+h, x: x+ w*2]
img["edge_croped"] =  imutils.resize(img["edge_croped"], height=1250)
img["orginal_croped"] = imutils.resize(img["orginal_croped"], height=1250)
cv.imwrite('edge_croped.jpg', img["edge_croped"])
cv.imwrite('orginal_croped.jpg', img["orginal_croped"])

contours, _ =
cv.findContours(img["edge_croped"],cv.RETR_TREE,cv.CHAIN_APPROX_SIMPLE)
c = cv.drawContours(img["edge_croped"], contours, -1, (150,150,150), 3)

cv.imwrite('show.jpg', c)
contour_areas = []
con = []
for contour in contours:
    area = cv.contourArea(contour)
    contour_areas.append(area)


mean = np.array(contour_areas).mean()

prec = [-1, -1]
points = []

for i, contour in enumerate(contours):
    epsilon = 0.1*cv.arcLength(contour,True)
    x,y,w,h = cv.boundingRect(contour)
    if(prec[0] == x and prec[1] == y):
        continue
    prec = [x, y]
    area = w * h

    if mean > area and area > 1000:
        if len(points) > 0 and destance(points[-1][0],points[-1][1], x, y) < 5:
            continue
        points.append((x, y, w, h))
print(len(points))
301
```

In [18]:

```
print(len(points))
for point in points:
    img["orginal_croped"] = cv.circle(img["orginal_croped"],
(point[0],point[1]), radius=12, color=(101, 111, 111))
plt.imshow(img["orginal_croped"], 'gray')
cv.imwrite('show.jpg', img["orginal_croped"])
301
```

Out[18]:

```
True
```

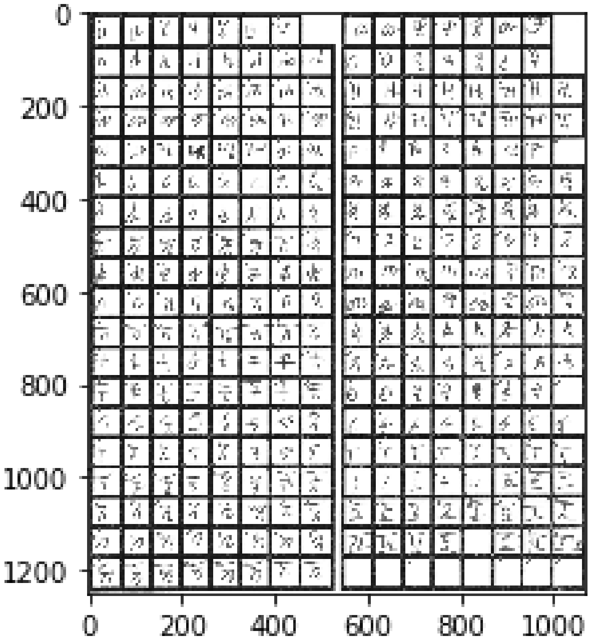


Figure 35

In [19]:

```
plt.imshow(img["orginal_croped"], 'gray')
print(img["orginal_croped"].shape)
(1250, 1066)
```

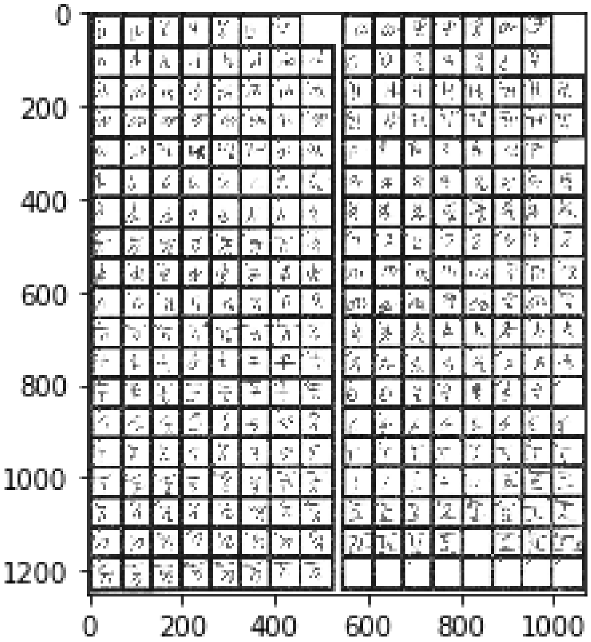


Figure 36

In [20]:

```
def is_empty(img):
    contours, _ = cv.findContours(img,cv.RETR_TREE,cv.CHAIN_APPROX_SIMPLE)
    max_contor = list(map(cv.contourArea, contours))
#      x, y, w, h = cv.boundingRect(max_contor)
    print(len(max_contor))
```

We got the cordinates of each boxes containing characters. Now we can crop each image, label and save.

In [21]:

```
sorted_points = _sort(points)
```

```
plt.imshow(img["orginal_croped"])
i = -1
count = 0;
c = 1

for rows in sorted_points:
    if len(rows):
        i += 1

#     print(i, len(rows))

    for j, point in enumerate(rows):
        x = point[0]
        y = point[1]
        w = point[2]
        h = point[3]

        if not i == 0 and j == 7:
            continue
        if j == 15:
            continue
        if i >= 15 and j > 7:
            continue

        crop = img["orginal_croped"][y: y+h+2, x: x+w]
        if len(points) == 299:
            if (i == 12 and j == 15) or (i == 18 and j > 7):
                continue
        elif len(points) == 301:
            if (i == 4 and j == 15) or (i == 12 and j == 15) or(i == 17 and j ==
12) or (i == 18 and j > 7):
                continue

#         cv.imwrite("sliced-images/"+str(i)+"x"+str(j)+".jpg", crop)
        cv.imwrite("sliced-images/"+str(c)+".jpg", crop)
        c += 1
        if i == 13 and j == 14 and len(points) == 299:
            c += 1
```

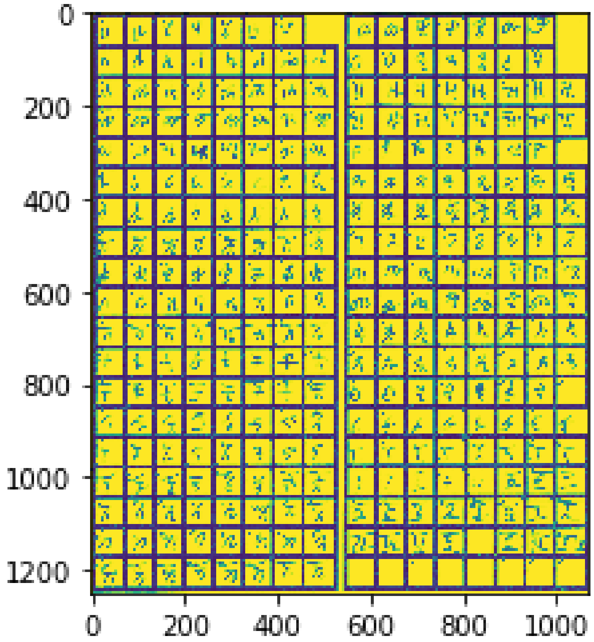


Figure 37

4.3.2. Extract characters

In the previous algorithm we have sliced the images, but there is still one problem. Characters are positioned in different manner, some are on the center bottom some are on bottom left or right. There for we need to crop out and center only the character

In [15]:

```
from matplotlib import pyplot as plt
import cv2 as cv
import numpy as np
import imutils
```

In [2]:

```
img = cv.imread('/home/eba/Final project/geez-caracter-recognition/dataset/sliced-images-liner-reduced/MX-M464N_20200211_205448_082/204.jpg')
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
original_img = img.copy()
plt.imshow(img)
plt.show()
```

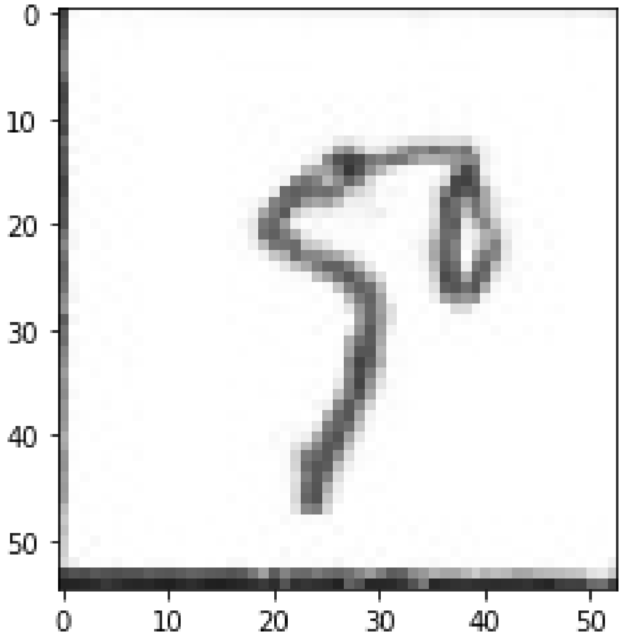


Figure 38

In [3]:

```
l = np.array([20,20,20])
h = np.array([250,250,250])
mask = cv.inRange(img, l, h)

plt.imshow(mask, 'gray')
```

Out[3]:

```
<matplotlib.image.AxesImage at 0x7fca8326dcf8>
```

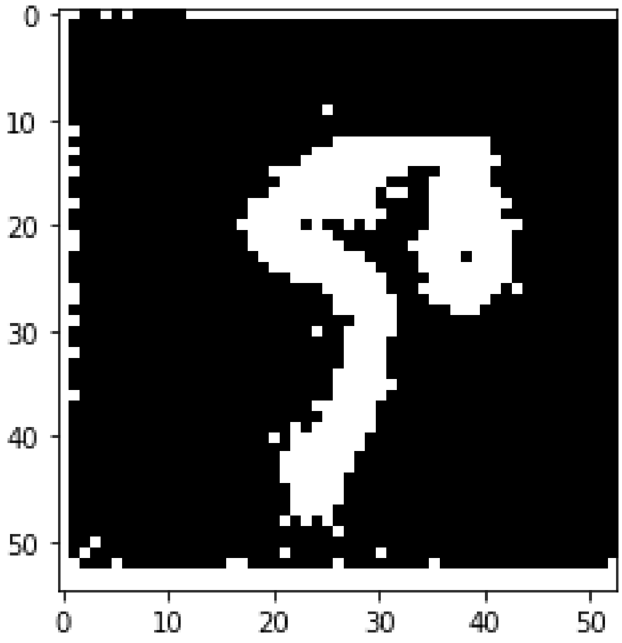


Figure 39

In [4]:

```
horizontal_kernel = cv.getStructuringElement(cv.MORPH_RECT, (10,1))
detected_lines = cv.morphologyEx(mask, cv.MORPH_OPEN, horizontal_kernel,
iterations=2)
plt.imshow(detected_lines)
plt.show()
```

```
contours, _ =
cv.findContours(detected_lines,cv.RETR_TREE,cv.CHAIN_APPROX_SIMPLE)
if len(contours) > 0:
    max_contor = max(contours, key = cv.contourArea)
    x, y, w, h = cv.boundingRect(max_contor)
    img = cv.rectangle(img, (x, y), (x+w, y+h), (255,255,255), -1)

plt.imshow(img)
```

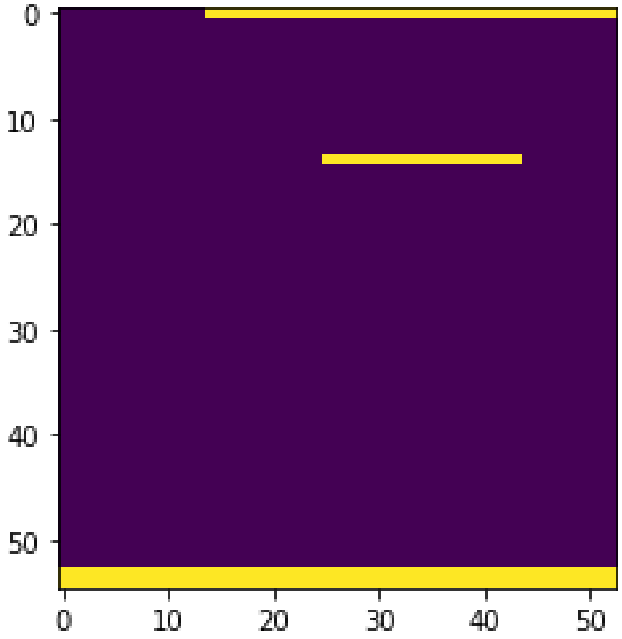


Figure 40

Out[4]:

```
<matplotlib.image.AxesImage at 0x7fca831c3358>
```

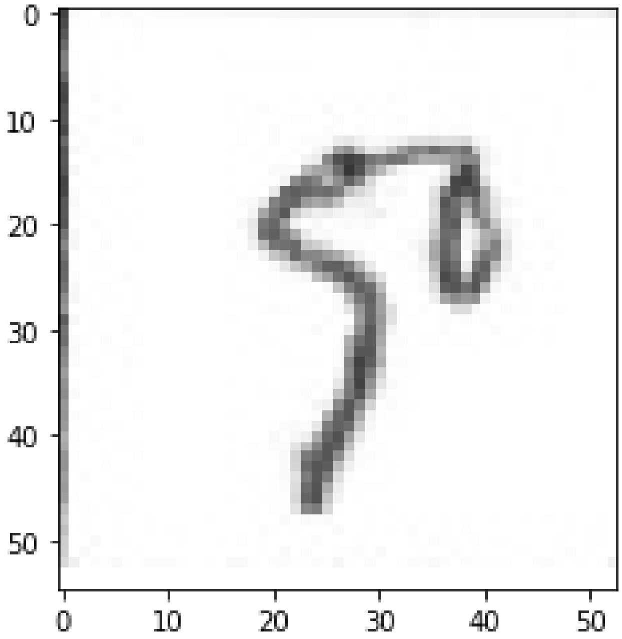


Figure 41

In [5]:

```
l = np.array([30,30,30])
h = np.array([250,250,250])
```

```
mask = cv.inRange(img, l, h)

plt.imshow(mask, 'gray')
```

Out[5]:

```
<matplotlib.image.AxesImage at 0x7fca831199b0>
```

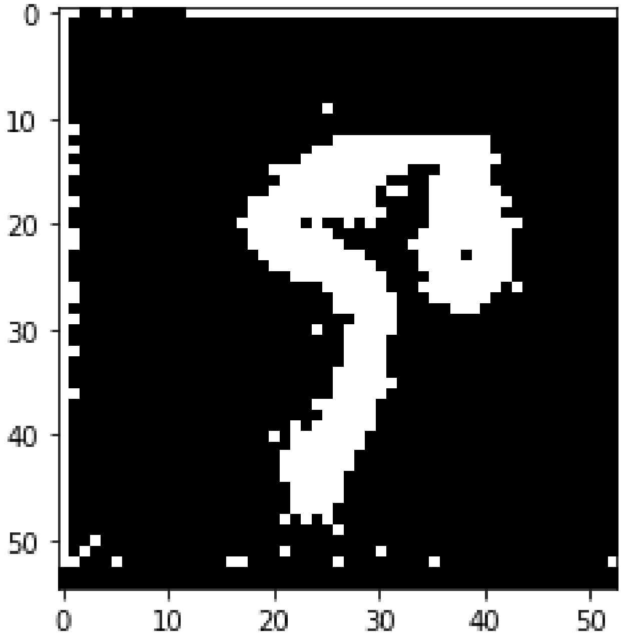


Figure 42

In [6]:

```
cont, _ = cv.findContours(mask, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_NONE)
```

In [7]:

```
cont_img = cv.drawContours(img, cont, -1, 255, 1)
```

In [8]:

```
plt.imshow(cont_img)
```

Out[8]:

```
<matplotlib.image.AxesImage at 0x7fca830ff518>
```

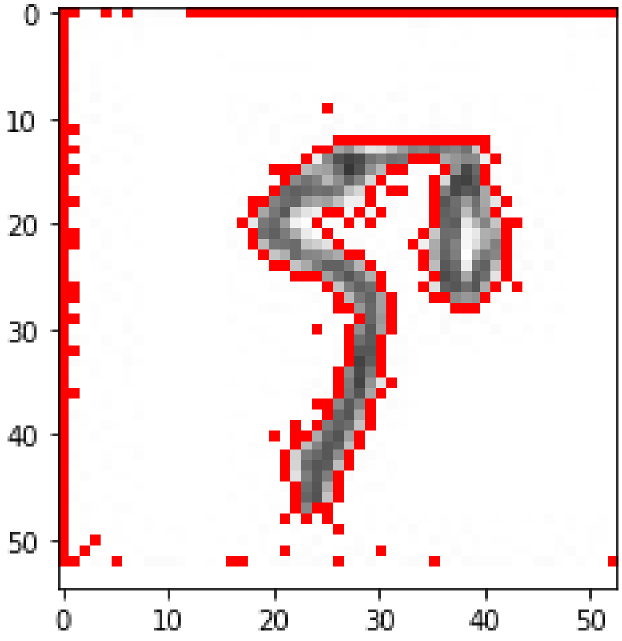


Figure 43

In [9]:

```
c = max(cont, key = cv.contourArea)
cont.sort(key = cv.contourArea)
if len(cont) > 0:
    x1, y1, w1, h1 = cv.boundingRect(cont[-1])
    print(x1, y1, w1, h1)
    print(len(cont))
17 12 27 38
15
```

In [10]:

```
# cv.rectangle(img, (x,y), (x+w, y+h), (0, 255, 0), 1)
cv.rectangle(img, (x1,y1), (x1+w1, y1+h1), (255, 255, 0), 1)
x, y, w, h = x1, y1, w1, h1
if len(cont) > 1 and cv.contourArea(cont[-2]) > 20:
    x2, y2, w2, h2 = cv.boundingRect(cont[-2])
    cv.rectangle(img, (x2,y2), (x2+w2, y2+h2), (255, 0, 255), 1)
    cv.rectangle(img, (min(x1, x2), min(y1, y2)), (max(x1+w1, x2+w2), max(y1+h1,
y2+h2)), (0, 255, 255), 1)
    x, y, x2, y2 = min(x1, x2), min(y1, y2), max(x1+w1, x2+w2), max(y1+h1,
y2+h2)
    x, y, w, h = x, y, x2 - x, y2 - y
    print(x, y, w, h)
    print(img.shape)
plt.imshow(img)
```

Out[10]:

```
<matplotlib.image.AxesImage at 0x7fca8306ac50>
```

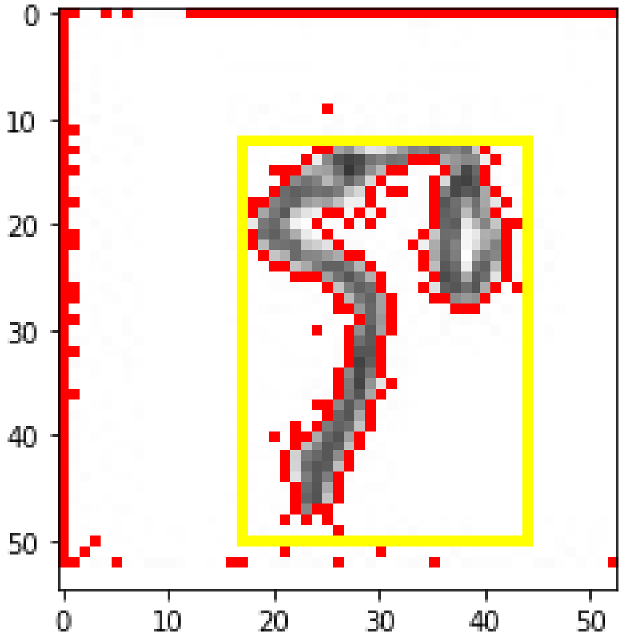


Figure 44

In [11]:

```
croped_img = original_img[y:y+h, x:x+w]
# croped_img = np.pad(croped_img, ((5, 5), (5, 5)), 'constant',
constant_values=(255, 255, 255))
plt.imshow(croped_img)
```

Out[11]:

```
<matplotlib.image.AxesImage at 0x7fca830439b0>
```

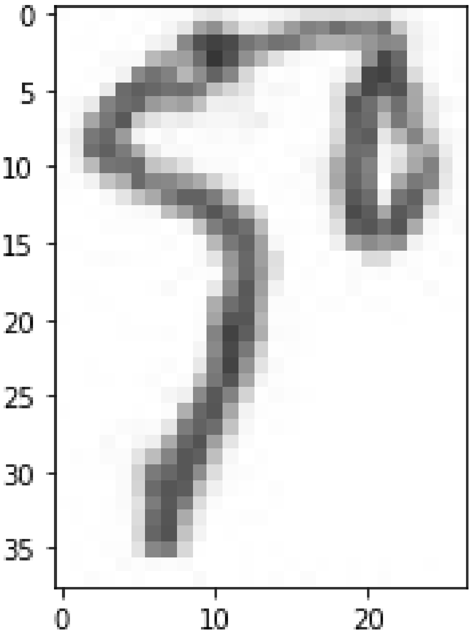


Figure 45

In [12]:

```
img_gray = cv.cvtColor(croped_img, cv.COLOR_BGR2GRAY)
```

In [16]:

```
 y_h, x_w = img_gray.shape
try:
    img_resized = imutils.resize(img_gray, height=28) if y_h > x_w else
imutils.resize(img_gray, width=28)
except Exception as e:
    print(e)

if y_h > x_w:
    diff = abs(img_resized.shape[0] - img_resized.shape[1])
    padd = (diff) // 2
    padding = (padd + 2, padd + 2 if diff % 2==0 else padd + 3)
    sqr_img = np.pad(img_resized, ((2, 2), padding), 'constant',
constant_values=(255, 255))
else:
    diff = abs(img_resized.shape[1] - img_resized.shape[0])
    padd = (diff) // 2
    padding = (padd + 2, padd + 2 if diff % 2 == 0 else padd + 3)
    sqr_img = np.pad(img_resized, (padding, (2, 2)), 'constant',
constant_values=(255, 255))
```

In [ ]:

```
gusian_blur = cv.GaussianBlur(sqr_img,(3,3),0)
# plt.imshow(gusian_blur, 'gray')
# ret,threshold_img = cv.threshold(gusian_blur,220,255,cv.THRESH_BINARY_INV)
# gusian_blur = cv.GaussianBlur(threshold_img,(3,3),0)
plt.imshow(gusian_blur, 'gray')
```

In [ ]:

```
print((list(map(min, gusian_blur))))
```

In [ ]:

```
alpha, beta = 1.1, 0
adjusted = cv.convertScaleAbs(gusian_blur, alpha=alpha, beta=beta)
plt.imshow(adjusted, 'gray')
```

4.3.3. Training the model

In [ ]:

```
import keras
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, Flatten, Conv2D,
MaxPooling2D
from keras.layers.normalization import BatchNormalization
import numpy as np
import h5py
import matplotlib.pyplot as plt
```

In [ ]:

```
np.random.seed(1000)
```

In [ ]:

```
model = Sequential()
```

In [ ]:

```
model = Sequential()

#1st Convolutional Layer
model.add(Conv2D(filters=96, input_shape=(32,32,3), kernel_size=(11,11),
strides=(4,4), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))

#2nd Convolutional Layer
model.add(Conv2D(filters=256, kernel_size=(5, 5), strides=(1,1),
padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))

#3rd Convolutional Layer
model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))

#4th Convolutional Layer
model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))

#5th Convolutional Layer
model.add(Conv2D(filters=256, kernel_size=(3,3), strides=(1,1), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))

#Passing it to a Fully Connected layer
model.add(Flatten())
# 1st Fully Connected Layer
model.add(Dense(4096, input_shape=(32,32,3,)))
model.add(BatchNormalization())
model.add(Activation('relu'))
# Add Dropout to prevent overfitting
model.add(Dropout(0.4))

#2nd Fully Connected Layer
```

```
model.add(Dense(4096))
model.add(BatchNormalization())
model.add(Activation('relu'))
#Add Dropout
model.add(Dropout(0.4))

#3rd Fully Connected Layer
model.add(Dense(1000))
model.add(BatchNormalization())
model.add(Activation('relu'))
#Add Dropout
model.add(Dropout(0.4))

#Output Layer
model.add(Dense(240))
model.add(BatchNormalization())
model.add(Activation('softmax'))
```

In [ ]:

```
#Model Summary
model.summary()
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_5 (Conv2D)            (None, 8, 8, 96)          34944     
_________________________________________________________________
batch_normalization_9 (Batch (None, 8, 8, 96)          384       
_________________________________________________________________
activation_9 (Activation)    (None, 8, 8, 96)          0         
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 96)          0         
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 4, 4, 256)         614656    
_________________________________________________________________
batch_normalization_10 (Batc (None, 4, 4, 256)         1024      
_________________________________________________________________
activation_10 (Activation)   (None, 4, 4, 256)         0         
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 2, 2, 256)         0         
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 2, 2, 384)         885120    
_________________________________________________________________
batch_normalization_11 (Batc (None, 2, 2, 384)         1536      
_________________________________________________________________
activation_11 (Activation)   (None, 2, 2, 384)         0         
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 2, 2, 384)         1327488   
_________________________________________________________________
batch_normalization_12 (Batc (None, 2, 2, 384)         1536      
_________________________________________________________________
activation_12 (Activation)   (None, 2, 2, 384)         0         
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 2, 2, 256)         884992    
_________________________________________________________________
batch_normalization_13 (Batc (None, 2, 2, 256)         1024      
_________________________________________________________________
activation_13 (Activation)   (None, 2, 2, 256)         0         
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 1, 256)         0         
_________________________________________________________________
```

```
flatten_1 (Flatten)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 4096)              1052672
_________________________________________________________________
batch_normalization_14 (Batc (None, 4096)              16384
_________________________________________________________________
activation_14 (Activation)   (None, 4096)              0
_________________________________________________________________
dropout_3 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_5 (Dense)              (None, 4096)              16781312
_________________________________________________________________
batch_normalization_15 (Batc (None, 4096)              16384
_________________________________________________________________
activation_15 (Activation)   (None, 4096)              0
_________________________________________________________________
dropout_4 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_6 (Dense)              (None, 1000)              4097000
_________________________________________________________________
batch_normalization_16 (Batc (None, 1000)              4000
_________________________________________________________________
activation_16 (Activation)   (None, 1000)              0
_________________________________________________________________
dropout_5 (Dropout)          (None, 1000)              0
_________________________________________________________________
dense_7 (Dense)              (None, 240)               240240
_________________________________________________________________
batch_normalization_17 (Batc (None, 240)               960
_________________________________________________________________
activation_17 (Activation)   (None, 240)               0
=================================================================
Total params: 25,961,656
Trainable params: 25,940,040
Non-trainable params: 21,616
_________________________________________________________________
```

In [ ]:

```
model.compile(loss = keras.losses.sparse_categorical_crossentropy, optimizer=
'adam', metrics=['accuracy'])
```

In [ ]:

```
# h5 = h5py.File('drive/MyDrive/Final project/dataset.h5', 'r')
# X = h5['dataset_1'][:]
X = np.load('drive/MyDrive/Final project/x.npy')
Y = np.load('drive/MyDrive/Final project/y.npy')
```

In [ ]:

```
from sklearn.model_selection import train_test_split
X_trn, X_tst, y_trn, y_tst = train_test_split(X, Y, test_size=0.2,
random_state=42)
y_tst[0] =  298
```

In [ ]:

```
from sklearn.utils.multiclass import unique_labels
from keras.utils import to_categorical
```

In [ ]:

```
# y_trn=to_categorical(y_trn)
# y_tst=to_categorical(y_tst)
```

In [ ]:

```
import cv2 as cv
def to_rgb(image):
      return cv.bitwise_not(cv.cvtColor(image, cv.COLOR_GRAY2BGR)) / 255
```

```
X_trn_rgb = np.array(list(map(to_rgb, X_trn)))
X_tst_rgb = np.array(list(map(to_rgb, X_tst)))

print(X_trn_rgb.shape)
(47100, 32, 32, 3)
```

In [ ]:

In [ ]:

```
from keras.preprocessing.image import ImageDataGenerator
train_generator = ImageDataGenerator(rotation_range=2,
horizontal_flip=True,zoom_range=.1 )
test_generator = ImageDataGenerator(rotation_range=2, horizontal_flip=
True,zoom_range=.1)
train_generator.fit(X_trn_rgb)
test_generator.fit(X_tst_rgb)
```

In [ ]:

```
from keras.callbacks import ReduceLROnPlateau
lrr= ReduceLROnPlateau(   monitor='val_acc',   factor=.01,   patience=3,
min_lr=1e-5)
```

In [ ]:

```
#Defining the parameters
batch_size= 100
epochs=100
learn_rate=.001
```

In [ ]:

```
#Training the model
history = model.fit_generator(train_generator.flow(X_trn_rgb, y_trn,
batch_size=batch_size), epochs = epochs, steps_per_epoch =
X_trn_rgb.shape[0]//batch_size, validation_data = test_generator.flow(X_tst_rgb,
y_tst, batch_size=batch_size), validation_steps = 250, callbacks = [lrr],
verbose=1)
/usr/local/lib/python3.6/dist-
packages/tensorflow/python/keras/engine/training.py:1844: UserWarning:
`Model.fit_generator` is deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/100
471/471 [==============================] - ETA: 0s - loss: 5.2794 - accuracy:
0.0192WARNING:tensorflow:Your input ran out of data; interrupting training. Make
sure that your dataset or generator can generate at least `steps_per_epoch *
epochs` batches (in this case, 250 batches). You may need to use the repeat()
function when building your dataset.
471/471 [==============================] - 38s 63ms/step - loss: 5.2783 -
accuracy: 0.0192 - val_loss: nan - val_accuracy: 0.0352
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr
Epoch 2/100
471/471 [==============================] - 24s 51ms/step - loss: 3.8932 -
accuracy: 0.1249
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 3/100
471/471 [==============================] - 24s 50ms/step - loss: 3.3890 -
accuracy: 0.1950
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 4/100
```

```
471/471 [==============================] - 23s 50ms/step - loss: 3.0377 -
accuracy: 0.2538
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 5/100
471/471 [==============================] - 23s 49ms/step - loss: 2.7834 -
accuracy: 0.3023
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 6/100
471/471 [==============================] - 24s 50ms/step - loss: 2.6204 -
accuracy: 0.3371
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 7/100
471/471 [==============================] - 24s 51ms/step - loss: 2.4888 -
accuracy: 0.3654
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 8/100
471/471 [==============================] - 24s 50ms/step - loss: 2.3694 -
accuracy: 0.3946
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 9/100
471/471 [==============================] - 22s 47ms/step - loss: 2.2524 -
accuracy: 0.4189
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 10/100
471/471 [==============================] - 23s 48ms/step - loss: 2.1915 -
accuracy: 0.4355
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 11/100
471/471 [==============================] - 22s 47ms/step - loss: 2.1116 -
accuracy: 0.4539
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 12/100
471/471 [==============================] - 22s 46ms/step - loss: 2.0470 -
accuracy: 0.4686
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 13/100
471/471 [==============================] - 22s 47ms/step - loss: 1.9948 -
accuracy: 0.4804
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 14/100
471/471 [==============================] - 22s 47ms/step - loss: 1.9380 -
accuracy: 0.4954
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 15/100
471/471 [==============================] - 22s 47ms/step - loss: 1.8723 -
accuracy: 0.5091
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 16/100
```

```
471/471 [==============================] - 22s 46ms/step - loss: 1.8362 -
accuracy: 0.5191
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 17/100
471/471 [==============================] - 22s 46ms/step - loss: 1.7767 -
accuracy: 0.5334
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 18/100
471/471 [==============================] - 22s 47ms/step - loss: 1.7212 -
accuracy: 0.5471
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 19/100
471/471 [==============================] - 22s 47ms/step - loss: 1.6765 -
accuracy: 0.5594
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 20/100
471/471 [==============================] - 22s 46ms/step - loss: 1.6672 -
accuracy: 0.5615
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 21/100
471/471 [==============================] - 22s 46ms/step - loss: 1.6194 -
accuracy: 0.5743
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 22/100
471/471 [==============================] - 22s 47ms/step - loss: 1.5627 -
accuracy: 0.5863
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 23/100
471/471 [==============================] - 22s 46ms/step - loss: 1.5296 -
accuracy: 0.5921
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 24/100
471/471 [==============================] - 22s 46ms/step - loss: 1.4871 -
accuracy: 0.6061
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 25/100
471/471 [==============================] - 22s 46ms/step - loss: 1.4640 -
accuracy: 0.6102
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 26/100
471/471 [==============================] - 22s 46ms/step - loss: 1.4349 -
accuracy: 0.6138
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 27/100
471/471 [==============================] - 22s 46ms/step - loss: 1.4105 -
accuracy: 0.6203
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 28/100
```

```
471/471 [==============================] - 22s 47ms/step - loss: 1.3948 -
accuracy: 0.6257
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 29/100
471/471 [==============================] - 22s 47ms/step - loss: 1.3607 -
accuracy: 0.6343
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 30/100
471/471 [==============================] - 23s 48ms/step - loss: 1.3624 -
accuracy: 0.6333
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 31/100
471/471 [==============================] - 23s 48ms/step - loss: 1.3279 -
accuracy: 0.6425
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 32/100
471/471 [==============================] - 22s 47ms/step - loss: 1.2972 -
accuracy: 0.6489
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 33/100
471/471 [==============================] - 23s 48ms/step - loss: 1.2834 -
accuracy: 0.6544
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 34/100
471/471 [==============================] - 23s 50ms/step - loss: 1.2434 -
accuracy: 0.6624
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 35/100
471/471 [==============================] - 23s 48ms/step - loss: 1.2411 -
accuracy: 0.6627
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 36/100
471/471 [==============================] - 23s 49ms/step - loss: 1.2308 -
accuracy: 0.6664
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 37/100
471/471 [==============================] - 23s 49ms/step - loss: 1.1801 -
accuracy: 0.6788
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 38/100
471/471 [==============================] - 23s 50ms/step - loss: 1.1704 -
accuracy: 0.6767
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 39/100
471/471 [==============================] - 24s 50ms/step - loss: 1.1431 -
accuracy: 0.6885
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 40/100
```

```
471/471 [==============================] - 23s 49ms/step - loss: 1.1462 -
accuracy: 0.6835
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 41/100
471/471 [==============================] - 24s 51ms/step - loss: 1.1325 -
accuracy: 0.6902
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 42/100
471/471 [==============================] - 23s 50ms/step - loss: 1.1013 -
accuracy: 0.6941
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 43/100
471/471 [==============================] - 24s 50ms/step - loss: 1.0995 -
accuracy: 0.6973
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 44/100
471/471 [==============================] - 23s 48ms/step - loss: 1.0719 -
accuracy: 0.7041
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 45/100
471/471 [==============================] - 23s 48ms/step - loss: 1.0542 -
accuracy: 0.7089
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 46/100
471/471 [==============================] - 23s 48ms/step - loss: 1.0637 -
accuracy: 0.7049
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 47/100
471/471 [==============================] - 24s 50ms/step - loss: 1.0295 -
accuracy: 0.7154
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 48/100
471/471 [==============================] - 24s 50ms/step - loss: 1.0113 -
accuracy: 0.7203
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 49/100
471/471 [==============================] - 22s 48ms/step - loss: 1.0011 -
accuracy: 0.7223
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 50/100
471/471 [==============================] - 23s 49ms/step - loss: 0.9869 -
accuracy: 0.7235
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 51/100
471/471 [==============================] - 22s 47ms/step - loss: 0.9826 -
accuracy: 0.7250
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 52/100
```

```
471/471 [==============================] - 22s 48ms/step - loss: 0.9679 -
accuracy: 0.7292
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 53/100
471/471 [==============================] - 22s 48ms/step - loss: 0.9446 -
accuracy: 0.7362
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 54/100
471/471 [==============================] - 22s 47ms/step - loss: 0.9488 -
accuracy: 0.7345
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 55/100
471/471 [==============================] - 22s 47ms/step - loss: 0.9288 -
accuracy: 0.7374
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 56/100
471/471 [==============================] - 22s 47ms/step - loss: 0.9198 -
accuracy: 0.7419
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 57/100
471/471 [==============================] - 22s 46ms/step - loss: 0.8947 -
accuracy: 0.7514
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 58/100
471/471 [==============================] - 21s 45ms/step - loss: 0.9057 -
accuracy: 0.7503
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 59/100
471/471 [==============================] - 24s 51ms/step - loss: 0.8890 -
accuracy: 0.7530
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 60/100
471/471 [==============================] - 22s 46ms/step - loss: 0.8742 -
accuracy: 0.7527
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 61/100
471/471 [==============================] - 22s 47ms/step - loss: 0.8687 -
accuracy: 0.7574
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 62/100
471/471 [==============================] - 22s 46ms/step - loss: 0.8672 -
accuracy: 0.7583
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 63/100
471/471 [==============================] - 22s 47ms/step - loss: 0.8508 -
accuracy: 0.7595
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 64/100
```

```
471/471 [==============================] - 22s 46ms/step - loss: 0.8413 -
accuracy: 0.7619
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 65/100
471/471 [==============================] - 22s 46ms/step - loss: 0.8272 -
accuracy: 0.7672
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 66/100
471/471 [==============================] - 22s 46ms/step - loss: 0.8168 -
accuracy: 0.7687
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 67/100
471/471 [==============================] - 22s 46ms/step - loss: 0.8157 -
accuracy: 0.7688
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 68/100
471/471 [==============================] - 22s 47ms/step - loss: 0.7912 -
accuracy: 0.7755
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 69/100
471/471 [==============================] - 22s 46ms/step - loss: 0.7815 -
accuracy: 0.7816
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 70/100
471/471 [==============================] - 22s 46ms/step - loss: 0.7844 -
accuracy: 0.7784
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 71/100
471/471 [==============================] - 21s 46ms/step - loss: 0.7790 -
accuracy: 0.7821
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 72/100
471/471 [==============================] - 24s 52ms/step - loss: 0.7579 -
accuracy: 0.7866
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 73/100
471/471 [==============================] - 22s 47ms/step - loss: 0.7672 -
accuracy: 0.7863
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 74/100
471/471 [==============================] - 22s 46ms/step - loss: 0.7502 -
accuracy: 0.7887
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 75/100
471/471 [==============================] - 22s 47ms/step - loss: 0.7429 -
accuracy: 0.7890
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 76/100
```

```
471/471 [==============================] - 22s 47ms/step - loss: 0.7280 -
accuracy: 0.7947
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 77/100
471/471 [==============================] - 22s 47ms/step - loss: 0.7237 -
accuracy: 0.7945
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 78/100
471/471 [==============================] - 22s 47ms/step - loss: 0.7108 -
accuracy: 0.7957
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 79/100
471/471 [==============================] - 22s 47ms/step - loss: 0.6991 -
accuracy: 0.8001
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 80/100
471/471 [==============================] - 22s 47ms/step - loss: 0.7060 -
accuracy: 0.8000
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 81/100
471/471 [==============================] - 22s 48ms/step - loss: 0.7004 -
accuracy: 0.8015
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 82/100
471/471 [==============================] - 22s 47ms/step - loss: 0.6974 -
accuracy: 0.8028
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 83/100
471/471 [==============================] - 23s 48ms/step - loss: 0.6738 -
accuracy: 0.8066
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 84/100
471/471 [==============================] - 23s 48ms/step - loss: 0.6637 -
accuracy: 0.8115
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 85/100
471/471 [==============================] - 22s 46ms/step - loss: 0.6647 -
accuracy: 0.8092
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 86/100
471/471 [==============================] - 22s 46ms/step - loss: 0.6554 -
accuracy: 0.8123
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 87/100
471/471 [==============================] - 22s 47ms/step - loss: 0.6465 -
accuracy: 0.8135
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 88/100
```

```
471/471 [==============================] - 22s 46ms/step - loss: 0.6433 -
accuracy: 0.8150
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 89/100
471/471 [==============================] - 23s 48ms/step - loss: 0.6393 -
accuracy: 0.8166
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 90/100
471/471 [==============================] - 22s 47ms/step - loss: 0.6277 -
accuracy: 0.8210
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 91/100
471/471 [==============================] - 22s 46ms/step - loss: 0.6264 -
accuracy: 0.8217
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 92/100
471/471 [==============================] - 22s 47ms/step - loss: 0.6178 -
accuracy: 0.8232
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 93/100
471/471 [==============================] - 22s 47ms/step - loss: 0.6220 -
accuracy: 0.8234
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 94/100
471/471 [==============================] - 22s 46ms/step - loss: 0.6093 -
accuracy: 0.8261
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 95/100
471/471 [==============================] - 22s 46ms/step - loss: 0.5994 -
accuracy: 0.8280
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 96/100
471/471 [==============================] - 22s 46ms/step - loss: 0.5847 -
accuracy: 0.8321
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 97/100
471/471 [==============================] - 22s 47ms/step - loss: 0.5974 -
accuracy: 0.8285
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 98/100
471/471 [==============================] - 26s 54ms/step - loss: 0.5969 -
accuracy: 0.8293
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 99/100
471/471 [==============================] - 22s 46ms/step - loss: 0.5877 -
accuracy: 0.8320
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc`
which is not available. Available metrics are: loss,accuracy,lr
Epoch 100/100
```

```
471/471 [==============================] - 22s 46ms/step - loss: 0.5687 - 
accuracy: 0.8344
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` 
which is not available. Available metrics are: loss,accuracy,lr
```

In [ ]:

```
score = model.evaluate(X_tst_rgb, y_tst)
368/368 [==============================] - 2s 4ms/step - loss: nan - accuracy: 
0.5585
```

In [ ]:

```
model.save("model")
INFO:tensorflow:Assets written to: model/assets
```

In [ ]:

```
plt.imshow(X_trn_rgb[3344], 'gray')
```

Out[ ]:

```
<matplotlib.image.AxesImage at 0x7fa1851f0908>
```

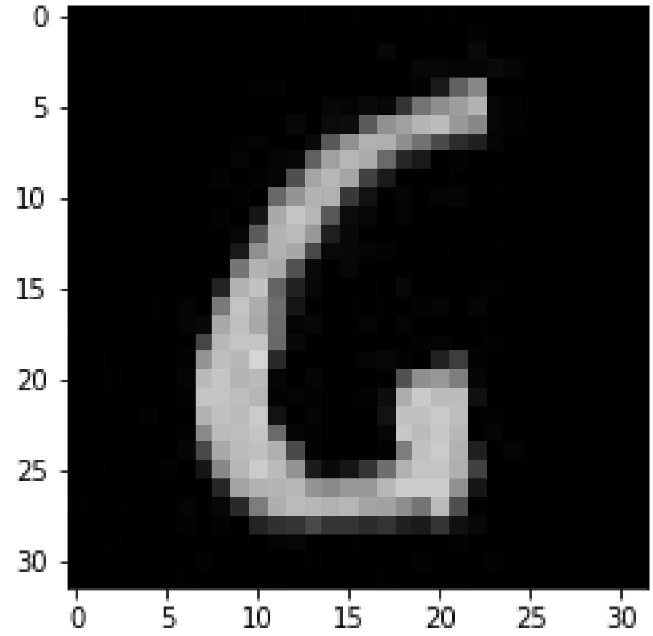


Figure 46

In [ ]:

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.savefig('alex-net-accurecy.png')
plt.show()
```

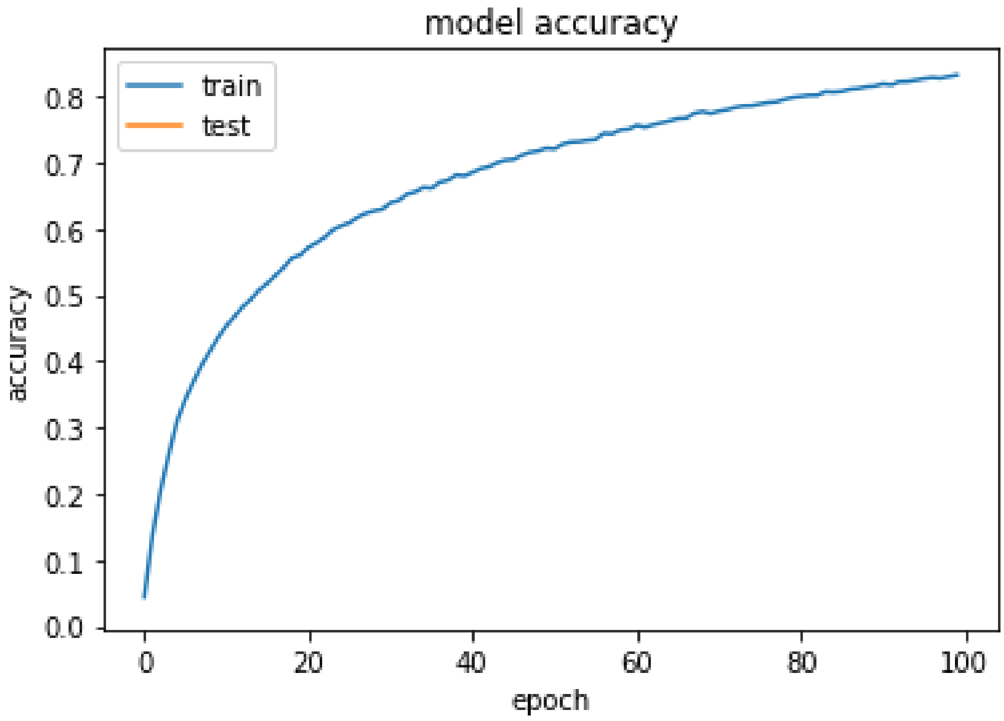


Figure 47

In [ ]:

```
plt.savefig('accurecy.png')
<Figure size 432x288 with 0 Axes>
```

In [ ]:

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.savefig('alex-net-loss.png')
plt.show()
```

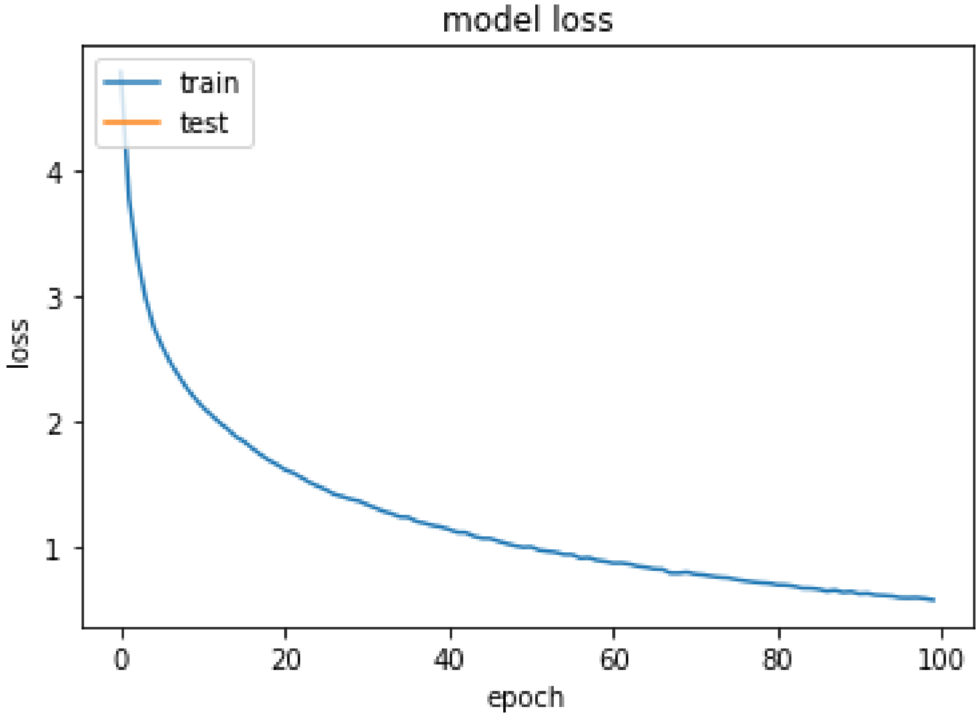


Figure 48

5. Testing

We have spited our dataset to 90 to 10 percent split for train and test set respectively. The test shows 83 – 88 percent accuracy.

6. Conclusion and Recommendations

6.1. Conclusion

One of the problems with deep learning technique is it require a lot of data to generalize and also bias on the training data is the other one. We managed to collect about 150,000 characters but labeling this data is the most challenging part of the project. There for we have just used 60,000 filtered and labeled data. But we managed to get good result. We have fond 83-88 % accuracy on training data. Though the model as a variance with some improvement it can be improved.

6.2. Recommendation and Future Enhancement

We recommend some techniques which are promising to fit geez characters dataset but we couldn’t train using those techniques.

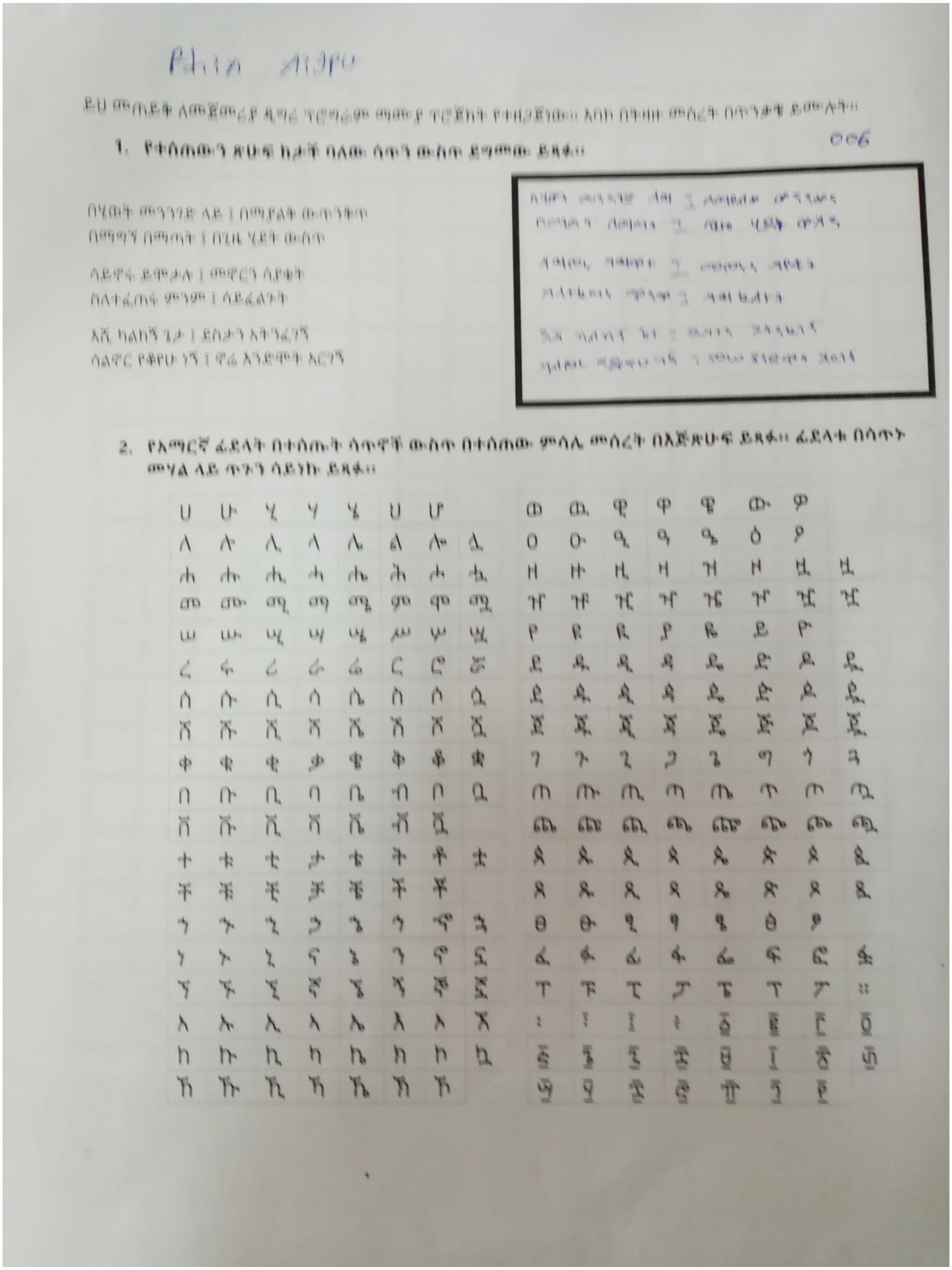
6.2.1. Transfer leaning

Transfer learning is one of the most promising techniques. It basically means using a model trained for other purpose for different use case. With this technique we think one can get more promising result with less data.

6.2.2. Multitask learning

Geez characters are two dimensional. We have a family of characters and a group of subtracts in them. Characters in a same family are very similar but different by a little future. There for training the neural network on this future may give good results. Multitask learning allow us to train neural network on two or more futures as the same time.

Appendices



[illegible]

ይህ መጠይቅ ለመጀመሪያ ዲግሪ ፕሮግራም ማሟያ ፕሮጀክት የተዘጋጀነው። እባክ በትዛዙ መሰረት በጥንቃቄ ይሙሉት።

006

1. የተሰጠውን ጽሁፍ ከታች ባለው ሳጥን ውስጥ ደግመው ይጻፉ።

በሂወት መንገድ ላይ ፤ በማያልቅ ውጥንቅጥ
በማግኘ በማጣት ፤ በጊዜ ሂደት ውስጥ

ሳይኖሩ ይሞታል ፤ መኖርን ሳያቁት
ስለተፈጠሩ ምንም ፤ ሳይፈልጉት

እሺ ካልከኝ ጌታ ፤ ደስታን አትንፈገኝ
ሳልኖር የቆየሁ ነኝ ፤ ኖሬ እንድሞት አርገኝ

በሂወት መንገድ ላይ ፤ በማያልቅ ውጥንቅጥ
በማግኘ በማጣት ፤ በጊዜ ሂደት ውስጥ

ሳይኖሩ ይሞታል ፤ መኖርን ሳያቁት
ስለተፈጠሩ ምንም ፤ ሳይፈልጉት

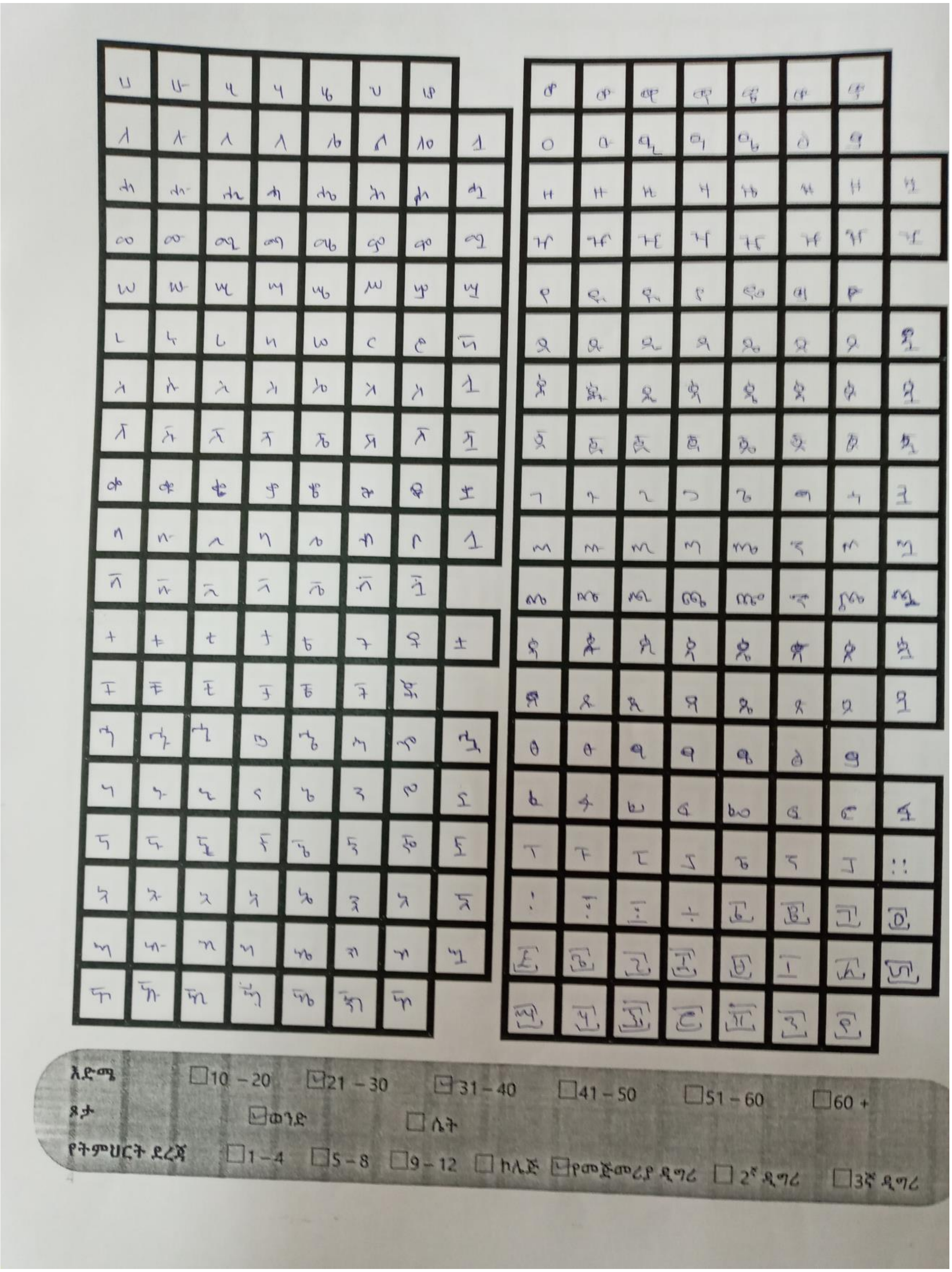
እሺ ካልከኝ ጌታ ፤ ደስታን አትንፈገኝ
ሳልኖር የቆየሁ ነኝ ፤ ኖሬ እንድሞት አርገኝ

2. የአማርኛ ፊደላት በተሰጡት ሳጥኖች ውስጥ በተሰጠው ምሳሌ መሰረት በእጅጽሁፍ ይጻፉ። ፊደላቱ በሳጥኑ መሃል ላይ ጥጉን ሳይነኩ ይጻፉ።

ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ	
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ	ሏ
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ	ሗ
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ	ሟ
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ	ሧ
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ	ሯ
ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ	ሷ
ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ	ሿ
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ	ቋ
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ	ቧ
ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቯ	
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ	ቷ
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ	
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ	ኋ
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ	ኗ
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ	ኟ
አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ	ኧ
ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ	ኳ
ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ	

ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ	
ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ	
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ	ዟ
ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ	ዧ
የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ	
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ	ዷ
ዸ	ዹ	ዺ	ዻ	ዼ	ዽ	ዾ	ዿ
ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ	ጇ
ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ	ጓ
ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ	ጧ
ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ	ጯ
ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ	ጷ
ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ	ጿ
ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ	
ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ	ፏ
ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ	።
፡	፣	፤	፥	፩	፪	፫	፬
፭	፮	፯	፰	፱	፲	፳	፴
፵	፶	፷	፸	፹	፺	፻	

Figure 49: Questioner first page



ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ	
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ	ሏ
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ	ሗ
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ	ሟ
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ	ሧ
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ	ሯ
ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ	ሷ
ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ	ሿ
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ	ቋ
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ	ቧ
ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ	
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ	ቷ
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ	
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ	ኋ
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ	ኗ
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ	ኟ
አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ	ኧ
ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ	ኳ
ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ	

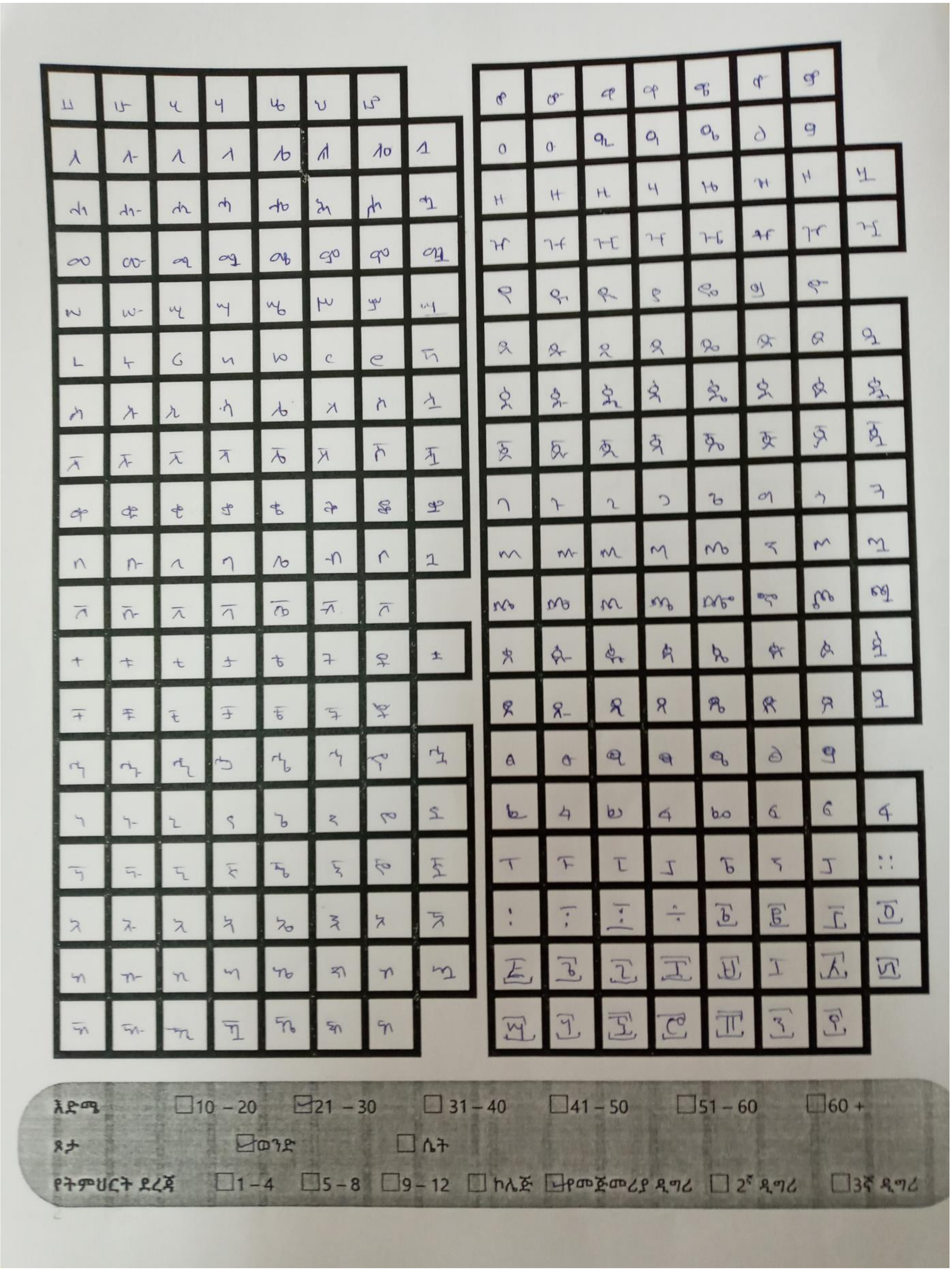
ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ	
ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ	
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ	ዟ
ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ	ዧ
የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ	
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ	ዷ
ዸ	ዹ	ዺ	ዻ	ዼ	ዽ	ዾ	ዿ
ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ	ጇ
ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ	ጓ
ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ	ጧ
ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ	ጯ
ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ	ጷ
ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ	ጿ
ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ	
ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ	ፏ
ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ	።
፡	፤	፦	፣	፩	፪	፫	፬
፭	፮	፯	፰	፱	፲	፳	፴
፵	፶	፷	፸	፹	፺	፻	

እድሜ □10 – 20 ☑21 – 30 ☑31 – 40 □41 – 50 □51 – 60 □60 +

ጾታ ☑ወንድ □ሴት

የትምህርት ደረጃ □1 – 4 □5 – 8 □9 – 12 □ኮሌጅ ☑የመጀመሪያ ዲግሪ □2ኛ ዲግሪ □3ኛ ዲግሪ

Figure 50: Questioner page 2 a



ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ	
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ	ሏ
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ	ሗ
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ	ሟ
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ	ሧ
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ	ሯ
ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ	ሷ
ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ	ሿ
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ	ቋ
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ	ቧ
ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ	
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ	ቷ
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ	
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ	ኋ
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ	ኗ
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ	ኟ
አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ	ኧ
ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ	ኳ
ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ	

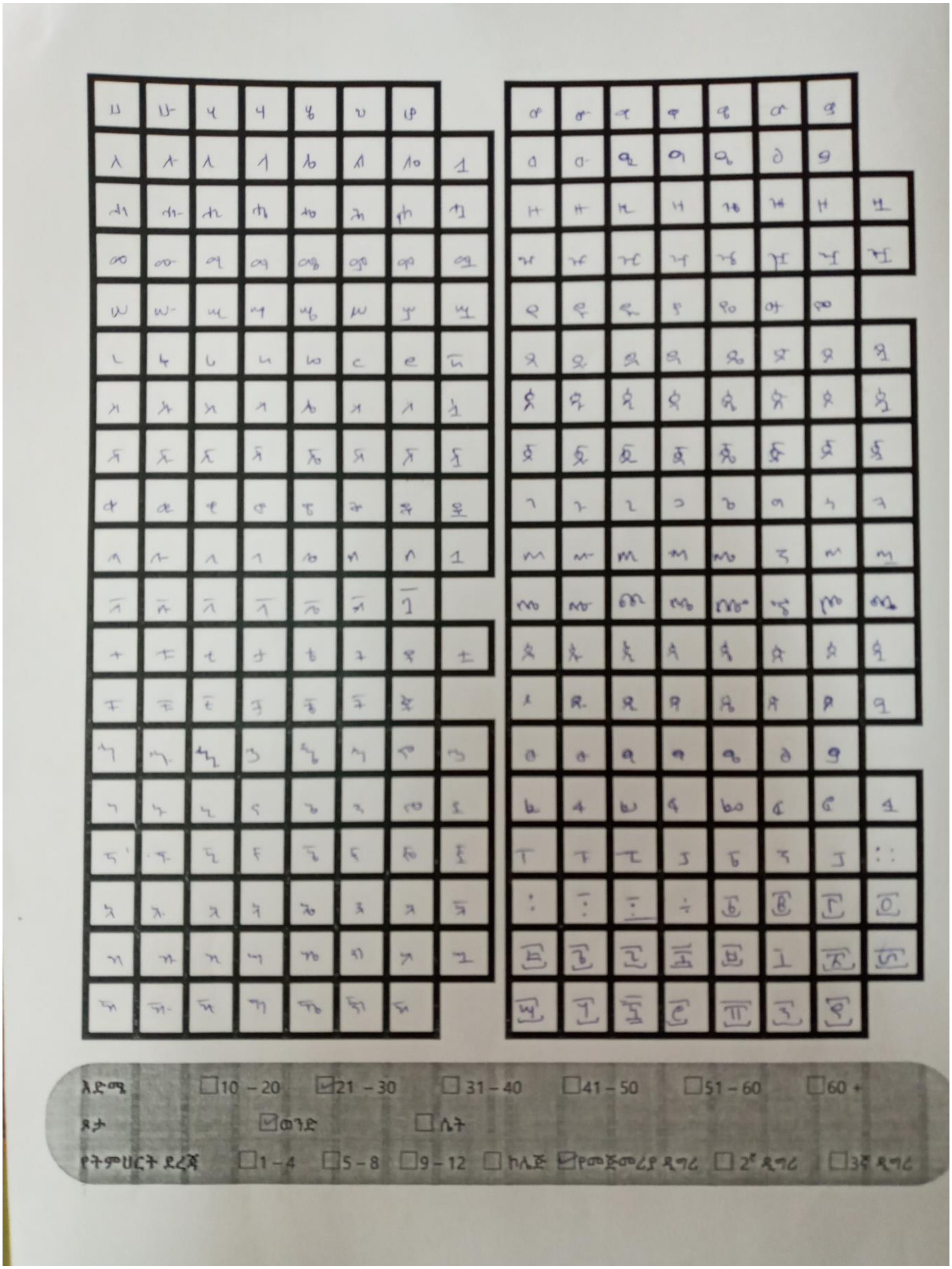
ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ	
ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ	
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ	ዟ
ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ	ዧ
የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ	
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ	ዷ
ዸ	ዹ	ዺ	ዻ	ዼ	ዽ	ዾ	ዿ
ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ	ጇ
ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ	ጓ
ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ	ጧ
ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ	ጯ
ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ	ጷ
ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ	ጿ
ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ	
ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ	ፏ
ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ	።
፡	፣	፤	፥	፩	፪	፫	፬
፭	፮	፯	፰	፱	፲	፳	፴
፵	፶	፷	፸	፹	፺	፻	

እድሜ □10 – 20 ☑21 – 30 □ 31 – 40 □41 – 50 □51 – 60 □60 +

ጾታ ☑ወንድ □ ሴት

የትምህርት ደረጃ □1 – 4 □5 – 8 □9 – 12 □ ኮሌጅ ☑የመጀመሪያ ዲግሪ □ 2ኛ ዲግሪ □3ኛ ዲግሪ

Figure 51: Questioner page 2b



ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ	
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ	ሏ
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ	ሗ
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ	ሟ
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ	ሧ
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ	ሯ
ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ	ሷ
ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ	ሿ
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ	ቋ
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ	ቧ
ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ	
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ	ቷ
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ	
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ	ኋ
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ	ኗ
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ	ኟ
አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ	ኧ
ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ	ኳ
ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ	

ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ	
ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ	
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ	ዟ
ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ	ዧ
የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ	
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ	ዷ
ዸ	ዹ	ዺ	ዻ	ዼ	ዽ	ዾ	ዿ
ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ	ጇ
ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ	ጓ
ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ	ጧ
ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ	ጯ
ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ	ጷ
ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ	ጿ
ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ	
ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ	ፏ
ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ	።
፡	፤	፥	፦	፩	፪	፫	፬
፭	፮	፯	፰	፱	፲	፳	፴
፵	፶	፷	፸	፹	፺	፻	

እድሜ ☐10 – 20 ☑21 – 30 ☐ 31 – 40 ☐41 – 50 ☐51 – 60 ☐60 +

ጾታ ☑ወንድ ☐ ሴት

የትምህርት ደረጃ ☐1 – 4 ☐5 – 8 ☐9 – 12 ☐ ኮሌጅ ☑የመጀመሪያ ዲግሪ ☐ 2ኛ ዲግሪ ☐3ኛ ዲግሪ

Figure 52: Questioner page 2 c

References

1. *Amharic character recognition using a fast signature based algorithm.* **Cowell, John and Fiaz Hussain.** s.l. : Proceedings on Seventh International Conference on Information Visualization, 2003. IV 2003., 2003.
2. *Offline handwritten Amharic word recognition.* **Yaregal Assabie a, Josef Bigun b.** 2019.
3. *Convolutional Neural Network Model for Arabic Handwritten Characters Recognition.* **Murtada Khalafallah Elbashir, Mohamed Elhafiz Mustafa.** 2018.
4. *Ethiopic Character Recognition Using Direction Field Tensor.* **Bigun, Yaregal Assabie and Josef.** 2006.
5. *Handwritten Amharic Character Recognition Using a Convolutional Neural Network.* **Mesay Samuel Gondere, Lars Schmidt, Abiot Sinamo Boltena, Hadi Samer Jomaa.** 2019.
6. *International Journal of Advanced Research in Computer Science and Software Engineering.* **Shaout, Omar Balola Ali and Adnan.**
7. *Off-Line Handwritten Character Recognition System Using Support Vector Machine.* **Gauri Katiyar1, Ankita Katiyar2, Shabana Mehfuz3.** 2017.
8. *Amharic Character Recognition using a Fast Signature Based Algorithm.* **Dr JOHN COWELL, Dr FIAZ HUSSAIN.** 2003.
9. *Building Fast and Compact Convolutional Neural Networks for Offline Handwritten Chinese Character Recognition.* **Xuefeng Xiaoa, Lianwen Jina,∗, Yafeng Yanga , Weixin Yanga, Jun Sunb , Tianhai Changa.** 2017.
10. *CNN based common approach to handwritten character recognition of multiple scripts.* **Sen Maitra, Durjoy, Bhattacharya, Ujjwal.** 2015.
11. *Amharic Character Recognition System For Printed Real-Life Documents.* **Birhanu, Abay Teshager.** 2010.
12. ImageNet Classification with deep convolutional neural networks. **Alex Krizhevsky, Ilya Sutskevr, Geoffery E. Hinton**