

**YILDIZ TECHNICAL UNIVERSITY**

**FACULTY OF CHEMISTRY METALLURGY**

**DEPARTMENT OF MATHEMATICAL ENGINEERING**

**MATHEMATICAL ENGINEERING GRADUATION PROJECT**

**Developing an android application for detecting mathematical equations using optical character recognition**

Advisor: Asist. Prof. Dr. Melih ÇINAR

Şaban Çepni

17058052

Istanbul, 2023

[1. INTRODUCTION 3](#_Toc136697403)

[1.1. Optical Character Recognition 3](#_Toc136697404)

[1.2. What is Tex and LaTex? 3](#_Toc136697405)

[1.3. What is Mathml? 4](#_Toc136697406)

[2. METHOD 5](#_Toc136697407)

[2.1. Image Processing 5](#_Toc136697408)

[2.2. Segmentation 7](#_Toc136697409)

[2.3. OCR of Single Character 9](#_Toc136697410)

[2.4. Summing into Latex 10](#_Toc136697411)

[2.5. Possible interpretation faults 11](#_Toc136697412)

[2.6. Wolfram Alpha API for solving 12](#_Toc136697413)

[3. CONCLUSION 14](#_Toc136697414)

[3.1. State of the Application 14](#_Toc136697415)

[3.2. How could application be improved 16](#_Toc136697416)

[APPENDIX 17](#_Toc136697417)

[REFERENCES 23](#_Toc136697418)

**LIST OF FIGURES**

1. Diagram 1: Simple application stages (Method)
2. Figure 1: Image Preprocessing [1] (Method - Image Processing)
3. Figure 2: Dividing image into rows and columns to isolate every single mathematical symbol [2] (Method - Segmentation)
4. Figure 3: Example for Confusion because of font (Method - OCR of Single Character)
5. Diagram 2: Complex diagram for how API’s and application interact with each other (Method)
6. Diagram 3: User interface and how to use (Method)

**FOREWORD**

I would like to thank my advisor, Assist. Prof. Dr. Melih CINAR, and my dear family for their patience and support.

**ABSTRACT**

The purpose of my application is to develop an optical character recognition (OCR) application that runs on the Android platform. This application offers the capability to solve mathematical equations using photographs. The working principle of the application has been explained in detail, outlining specific stages.

The optical character recognition section of the application is implemented using the pix2tex application's API. In this stage, an algorithm is used to detect and identify characters obtained from the photograph. Subsequently, the obtained characters are transformed into MathML format using the TexZilla API.

For equation solving, the Wolfram Alpha API is utilized. This API analyzes the equation from the photograph and provides accurate solutions. The application achieves a 92% success rate in accurate analysis and a 90% success rate in correct equation solving.

Thanks to these stages and the APIs used, the application enables users to solve mathematical equations with ease using photographs. The goal of the application is to assist users who want to solve mathematical equations conveniently.

**ÖZET**

Uygulamamın amacı, Android platformunda çalışan bir optik karakter tanıma (OCR) uygulaması geliştirmektir. Bu uygulama, matematiksel denklemleri fotoğraflarla çözümleyebilme özelliği sunmaktadır. Uygulamanın çalışma prensibi, belirli aşamaları içeren bir süreç üzerinden ayrıntılı bir şekilde açıklanmıştır.

Uygulamanın optik karakter tanıma bölümü, pix2tex uygulamasının API'sini kullanarak gerçekleştirilmiştir. Bu aşamada, fotoğraftan elde edilen karakterleri algılayan ve tanımlayan bir algoritma kullanılmaktadır. Daha sonra, elde edilen karakterler MathML formatına dönüştürülmek üzere TexZilla API'si kullanılmıştır.

Denklemi çözümlemek için ise Wolfram Alpha API'si kullanılmıştır. Bu API, fotoğraftaki denklemi analiz ederek doğru çözümleri sunmaktadır. Uygulama, %92 başarı oranıyla doğru analiz etme ve %90 başarı oranıyla doğru çözümleme sağlamaktadır.

Bu aşamalar ve kullanılan API'lar sayesinde uygulama, kullanıcılara matematiksel denklemleri fotoğraflarla çözme imkanı sunmaktadır. Uygulamanın amacı, matematiksel denklemleri kolaylıkla çözebilmek isteyen kullanıcılara yardımcı olmaktır.

## INTRODUCTION

Developing an Android app for math equation detection using OCR involves leveraging advanced image processing algorithms and OCR technology to automatically recognize and analyze mathematical equations from images. This application aims to provide users with the ability to capture or select an image containing mathematical content, extract the text using OCR algorithms, and process the detected equations for further analysis or solving. Such an app can be a valuable tool for students, mathematicians, and anyone working with mathematical equations.

First I’ll talk about the OCR then talk about the tools for stringing math equations LaTex and MathML.

* 1. **Optical Character Recognition**

Optical Character Recognition (OCR) is a technology that enables machines to identify and interpret characters using optical methods. It encompasses various technologies that translate scanned or captured images of text, whether handwritten or typewritten, into machine-readable text with the same meaning. The primary goal of OCR is to analyze the optical patterns present in an image and assign them to corresponding alphanumeric or other characters. Once OCR is performed, additional processing can be applied to the recognized text, such as converting it to speech. OCR is a research field within computer vision and artificial intelligence, focusing on the development of algorithms and techniques for character recognition.

Closed-Form Expression evaluation is considered one of the simpler tasks in mathematics. Similarly, developing a computer application for this purpose is also relatively straightforward. Computers equipped with built-in operators can easily handle many simple mathematical equations. However, as the complexity of equations increases, finding solutions becomes more challenging. This is primarily due to the requirement of more sophisticated problem-solving techniques and algorithmic procedures. Fortunately, numerous systems are available today that assist in solving complex mathematical equations. By combining these systems with Optical Character Recognition (OCR) technology, it becomes possible to tackle the computation of printed expressions in a new domain.

* 1. **What is Tex and LaTex?**

TeX is a typesetting system developed by Donald Knuth in the late 1970s. It is a low-level programming language that provides the foundational tools for typesetting and creating documents. TeX focuses on the precise positioning of characters, symbols, and elements on a page, ensuring high-quality typesetting.

LaTeX, on the other hand, is a set of macros built on top of TeX. It provides a higher-level, user-friendly interface for document preparation, especially for complex documents that include mathematical equations and technical content. LaTeX simplifies the process of typesetting by offering predefined commands and document classes that handle the formatting and layout automatically. It allows users to focus on the content rather than the details of typesetting. [Ref. 3]

Key points about LaTeX:

1. LaTeX is a typesetting system for documents with mathematical and technical content.
2. It uses a markup language to define the structure and formatting of the document.
3. LaTeX provides high-quality output and precise control over layout and formatting.
4. It is widely used in academia and scientific fields for research papers, theses, and books.
5. LaTeX requires a compilation process to generate the final document output.
   1. **What is Mathml?**

MathML (Mathematical Markup Language) is an XML-based markup language specifically designed for representing mathematical equations and formulas on the web. It allows for the accurate representation and communication of mathematical content across different platforms and devices. MathML defines a set of tags and attributes that describe the structure and presentation of mathematical expressions. It is supported by various web browsers and mathematical software tools, enabling the display and manipulation of mathematical content on the internet. [3]

Key points about MathML:

1. MathML is an XML-based markup language for representing mathematical equations on the web.
2. It provides a standard syntax for describing the structure and presentation of mathematical expressions.
3. MathML enables accurate representation and communication of mathematical content.
4. It is supported by web browsers and mathematical software tools.
5. MathML facilitates accessibility and interoperability of mathematical content on the internet.

## METHOD

In this project I used Android Studio Java, Pix2Tex API, Texzilla, Node.js for Texzilla, Python for pix2tex, Wolfram Alpha API and for API checking Postman. It could be done using less or more its preferation.

First of all I’ll show how application works using simple diagram then I’ll explain stage by stage how exactly it works.

metin, diyagram, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Diagram 1 Simple application stages

Until conversion to Wolfram Alpha Format all parts of OCR are done by the API Pix2Tex [6]. And all communication is done using the code shown in [Code 1]. API is currently working in my computer using python and using port forwarding I can connect to it from anywhere.

* 1. **Image Processing**

In the first stage of image processing, we will perform a series of operations to transform the image as follows:

1. **Convert the image to grayscale**: This involves removing the hue and saturation information from the image while retaining only the intensity. The resulting image will be a grayscale representation where the brightness of each pixel corresponds to its intensity.
2. **Convert the grayscale image to black and white**: Since we assume that the initial image consists of a background and a mathematical expression, we want to extract the characters and symbols by converting the grayscale image to black and white. This operation will result in a binary image where the background will be represented by white pixels and the characters and symbols will be represented by black pixels.
3. **Remove connected groups of pixels with less than 9 pixels**: In order to reduce noise and eliminate small artifacts, we will remove any connected groups of pixels that have less than 9 pixels. This operation helps in smoothing the image and removing any small isolated elements that might interfere with further processing.
4. **Crop the resulting image to the minimal bounding rectangle**: To focus solely on the mathematical expression and remove any redundant background, we will crop the resulting image to the minimal bounding rectangle. This rectangle will tightly enclose the expression while excluding any unnecessary surrounding areas.

By performing these operations, we aim to obtain an image that emphasizes the mathematical expression with a white background and black characters/symbols, while minimizing noise and extraneous information.

metin, yazı tipi, sayı, numara, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Fig. 1 Image Preprocessing [1]

* 1. **Segmentation**

To perform a segmentation on a given binary (black and white) image while preserving the correct order of characters in the expression it represents, we have defined the following terms:

**Valid Column:**

A valid column is a column in the image that has a height equal to the height of the image and a width of at least 1. Additionally, all the vertical unit-width columns within this valid column must contain at least one black pixel. In other words, it is a column that extends from the top to the bottom of the image and contains at least one vertical strip of black pixels.

**Maximal Valid Column:**

A maximal valid column is a valid column that is of the largest possible size. This means that if another unit-pixel column is added to either the left or right side of this valid column, it would become invalid because the added column would not contain any black pixels. Therefore, a maximal valid column is the largest continuous column in the image that satisfies the criteria of a valid column.

**Valid Row:**

A valid row is a row of pixels in the image that has a height of at least 1. Additionally, all the horizontal unit-width rows within this valid row must contain at least one black pixel. In simpler terms, it is a row that spans the entire width of the image and contains at least one horizontal strip of black pixels.

**Maximal Valid Row:**

A maximal valid row is a valid row that is of the largest possible size. This means that if another unit-pixel row is added to either the top or bottom of this valid row, it would become invalid because the added row would not contain any black pixels. Hence, a maximal valid row is the largest continuous row in the image that satisfies the criteria of a valid row.

**Connected Component:**

A connected component is a set of pixels in the image where each pixel has at least one neighboring pixel from the same set in any of the eight directions (North, Northwest, West, Southwest, South, Southeast, East, Northeast). In other words, all pixels in a connected component are connected to each other through adjacent positions.

**Clump:**

A clump is a connected component of maximal size in the image. Any pixel that is external to the clump would break the connectivity requirement if added to the clump. Dividing a black and white image into clumps is the simplest form of segmentation, where each clump represents a distinct character or symbol in the expression.

By defining these terms, we can use them as building blocks to segment the given binary image into individual characters, while maintaining the correct left-to-right and top-to-bottom order. The process of reversing the order of characters in special cases is done in stage 4 of the overall image processing pipeline. [2]

yazı tipi, metin, sayı, numara, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Fig. 2. Dividing image into rows and columns to isolate every single mathematical symbol. [2]

* 1. **OCR of Single Character**

Once the image has been processed and segmented into individual rows and columns, we can now proceed with the OCR of each isolated mathematical symbol or character. OCR is the process of recognizing and interpreting characters from an image.

To perform OCR on a single character, we follow these steps:

1. **Preprocessing:**

* Resize the character image to a standard size to ensure consistent recognition.
* Convert the character image to grayscale to remove any color information.
* Apply any necessary filters or enhancements, such as noise reduction or contrast adjustment, to improve character legibility.

1. **Feature Extraction:**

* Extract relevant features from the preprocessed character image that are distinctive for character recognition.
* Common features include stroke width, edge detection, contour properties, and pixel intensity distribution.
* These features serve as input for the character recognition algorithm.

1. **Character Recognition:**

* Utilize machine learning or pattern recognition algorithms to classify the character based on the extracted features.
* This can involve techniques such as template matching, neural networks, or support vector machines.
* Train the recognition model using a dataset of labeled characters to improve accuracy.

1. **Post-processing:**

* Apply post-processing techniques to refine the recognition results and correct any errors or inconsistencies.
* This may involve employing language-specific rules, dictionary lookups, or statistical analysis to improve the recognition accuracy.

By performing these steps, we can recognize and interpret each individual character or mathematical symbol within the expression image.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Fig. 3 Example for Confusion because of font

It is important to note that OCR accuracy can vary depending on factors such as image quality, font style, character complexity, and noise levels. Therefore, it is essential to optimize the preprocessing, feature extraction, and recognition stages to achieve the best possible accuracy for character recognition.

* 1. **Summing into Latex**

After successfully recognizing and interpreting the individual characters and symbols within the mathematical expression, the next step is to convert the detected equation into a LaTeX representation. LaTeX is a widely used typesetting system for mathematical expressions, providing a standard syntax for representing mathematical formulas.

To convert the recognized characters into LaTeX representation, we follow the following steps:

1. **Summing and Grouping:**

* Identify and group the recognized characters and symbols that form a coherent mathematical expression.
* Determine the correct order of operations (e.g., parentheses, exponents, multiplication, division, addition, subtraction) based on mathematical conventions.

1. **LaTeX Syntax Mapping:**

* Define a mapping between the recognized characters and their corresponding LaTeX representations.
* Create a lookup table that associates each character or symbol with the corresponding LaTeX command or notation.

1. **LaTeX Expression Generation:**

* Utilize the mapping and grouping information to generate the LaTeX representation of the mathematical expression.
* Assemble the recognized characters and symbols according to the correct order of operations and apply the corresponding LaTeX commands.

1. **Handling Special Cases:**

* Implement special handling for complex mathematical structures, such as fractions, square roots, matrices, and integrals.
* Adapt the LaTeX syntax generation to accommodate these special cases, ensuring accurate representation and proper formatting.
  1. **Possible interpretation faults**

This does make MathML and LaTex both necessary to use so I’m using Texzilla LaTex MathML viewer for conversation between them. Texzilla is turned to api using the node.js thats working my computer and for further connection port forwarding is done aswell.

In LaTeX and MathML, equations can sometimes be interpreted differently based on the context or the interpretation of symbols and operators. Reason why i’m using both of these types for solving equation is for some interpretation faults and wolfram api’s interpretation faults will talk about it after this. Some examples to these are;

**LaTeX Equations:**

* **Ambiguous Notation:** Certain LaTeX equations can be interpreted in multiple ways due to the lack of explicit grouping or parentheses. For example, \frac{1}{2}x can be read as "one-half times x" or "the fraction 1 divided by 2x."
* **Operator Precedence:** The interpretation of equations like 2 + 3 \times 4 can vary depending on the precedence given to addition and multiplication operators. Without clear grouping or parentheses, the result can differ.
* **Lack of Context:** Equations without additional context, such as x = y, can be ambiguous and have multiple possible meanings.
* **Notation Variations:** Different symbols or notations can be used to represent the same mathematical concept, leading to potential confusion.

**MathML Equations:**

* **Structural Differences:** MathML can have structural variations that impact interpretation, such as different nesting or ordering of elements, which can alter the meaning of the equation.
* **Presentation vs. Content MathML:** The distinction between Presentation MathML and Content MathML can lead to different interpretations. Presentation MathML focuses on visual rendering, while Content MathML emphasizes semantic meaning.
* **Lack of Standardization:** MathML does not enforce strict rules or conventions for certain mathematical expressions, resulting in varying interpretations among different MathML parsers or rendering engines.
  1. Wolfram Alpha API for solving

Till this point we got LaTex and MathML of our image for solving I’m using Wolfram Alpha API thats for free up to 2000 queries monthly.

The Wolfram Alpha API provides a powerful tool for solving equations in an academic setting. With its extensive computational knowledge base and advanced algorithms, it offers an efficient and reliable solution for solving a wide range of mathematical equations.

The API supports various equation types, including linear equations, quadratic equations, polynomial equations, trigonometric equations, and more. It can handle both single-variable and multi-variable equations, making it suitable for complex mathematical problems encountered in academic disciplines such as mathematics, physics, engineering, and economics. .[4]

I integrated Wolfram Alpha API into my OCR application for the purpose of solving equations.Wolfram Alpha API lets users input queries in mathml and latex but sometimes it could’not understand it in one of them because of the LaTex to MathML conversation issue, the type it takes MathML is needs to be simplified and sometimes its because of the LaTex and MathML interpretation faults.

Extra Diagrams for Application API interactions And How to use the application

**metin, diyagram, paralel, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Diagram 2 Complex diagram for how API’s and application interact with each other

metin, diyagram, çizgi, teknik çizim içeren bir resim

Açıklama otomatik olarak oluşturuldu

Diagram 3 User interface and how to use

## CONCLUSION

In this project i tried to create a reliable application for solving mathematical equations by performing OCR on mathematical expressions. One of the disadvantages of my system is that it requires calibration to read a specific font. The current version of the application works well on characters written in fonts similar to the one of the templates.

* 1. **State of the Application**

All of the tests done are for the current state of the application. After this I will say how can we improve all of the tests. Most of the times not being able to correctly analyze is because of the size of the image and symbol. Even when I use higher quality image its hard to get complex symbol as power of or upper lower limit of something. Because symbol gets even smaller and the distinctive details that define symbol gets smaller it makes it harder for analyzation of the symbol. So here are the results for now.

Tests done using the application:

| **Difficulty** | **Total Test Number** | **Correctly Analyzed Equation Number** | **Correctly Solved Equation Number** | **Correctly Analyzed Percentage** | **Correctly Solved Percentage** |
| --- | --- | --- | --- | --- | --- |
| Simple | 20 | 19 | 19 | 95% | 95% |
| Normal | 15 | 14 | 14 | 93.33% | 93.33% |
| Complex | 15 | 13 | 12 | 86.67% | 80% |
| Total | 50 | 46 | 45 | 92% | 90% |

As you can see we have correctly analyzed 92 percent of all equations. Keep in mind font picture quality and etc. Can effect this percentage greatly. Still its a high percentage. Correctly solved percentage is 2 percent lower than analyzed cause of the LaTex MathML faults.

Some Examples to Simple Normal Complex are;

| **Example** | **Analyzed Equation** | **Solution** |
| --- | --- | --- |
| metin, el yazısı, yazı tipi, yazı tahtası içeren bir resim  Açıklama otomatik olarak oluşturuldu |  | 1/160π^5 cos(2) |
| taslak, yazı tipi, diyagram, kırpıntı çizim içeren bir resim  Açıklama otomatik olarak oluşturuldu |  | 3 |
| el yazısı, metin, yazı tipi, hat sanatı, kaligrafi içeren bir resim  Açıklama otomatik olarak oluşturuldu |  | 2e^(√x) |
| siyah, karanlık içeren bir resim  Açıklama otomatik olarak oluşturuldu |  | Solution not found |
|  |  | 555 |

* 1. **How could application be improved**

There are so many API’s for OCR on math equations. One of the most promising is looking like mathpix. As a student I preferred to stay free. Having more up to date OCR with higher AI dataset would increase the analyzed percentage for sure. And having an conversation between LaTex to MathML sometimes have problems this and interpretation faults can be solved by getting your OCR to output exactly to your solving , API or applications input type would greatly increase the ones that get analyzed correctly to be solved. But even now most of the images can be solved using the application.

Adding preview of camera and selecting the equation part of the image feature could greatly increase the ease of using application. Thought about doing these and it could be done easily but would take some time.

## APPENDIX

Main Screen without Image

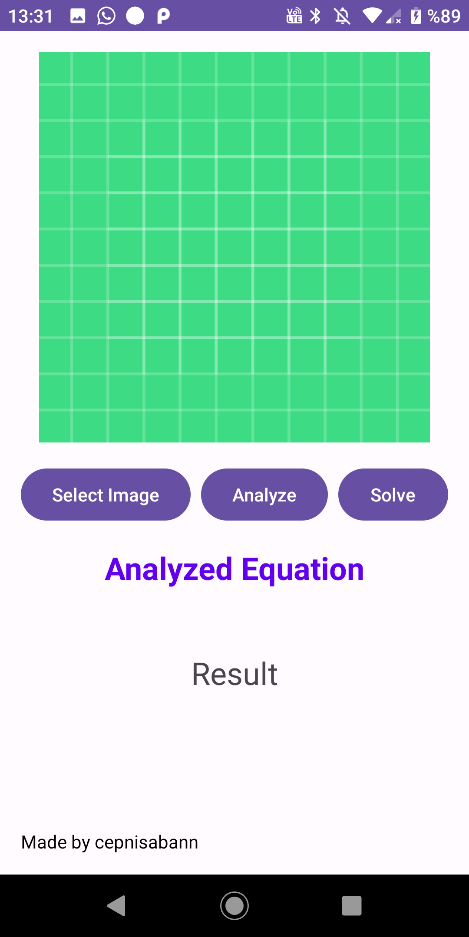
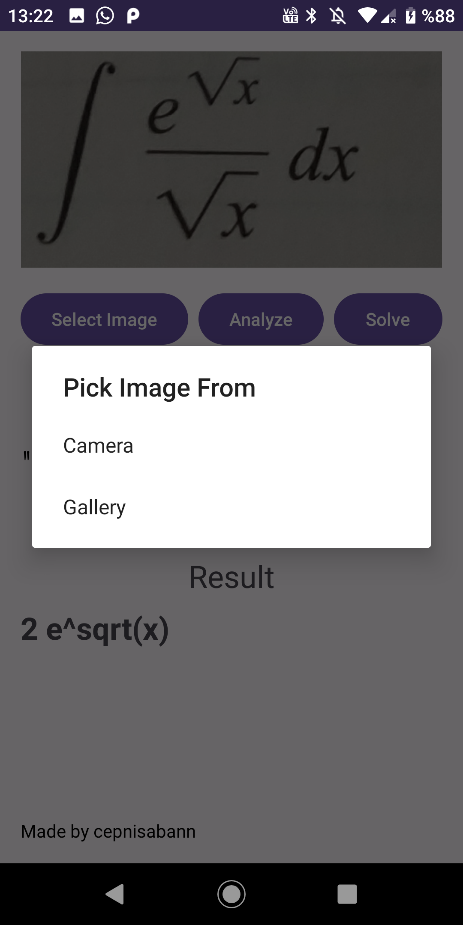
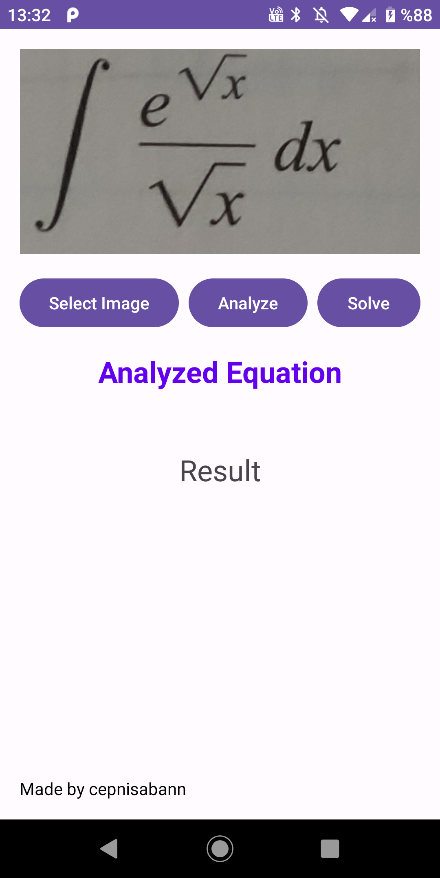


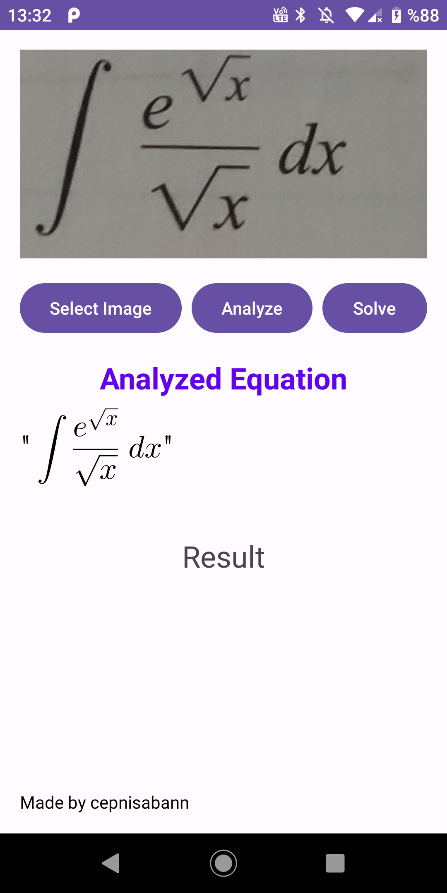
Image Selection



After Image Selected



After Analyze Button



Analyze API and after analyze showing equation code (Shortened)[Code 1]

private void convertImageToLaTeX(File imageFile) {  
 checkOnce = true;  
 OkHttpClient client = new OkHttpClient();  
 MediaType mediaType = MediaType.*parse*("image/jpeg");  
 RequestBody requestBody;  
 try {  
 requestBody = new MultipartBody.Builder()  
 .setType(MultipartBody.*FORM*)  
 .addFormDataPart("file", "image.jpg",  
 RequestBody.*create*(mediaType, imageFile))  
 .build();  
 } catch (Exception e) {  
 e.printStackTrace();  
 Toast.*makeText*(MainActivity.this, "Invalid file path", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
 Request request = new Request.Builder()  
 .url("http://159.146.11.126:8502/predict/") // API address should be set correctly  
 .post(requestBody)  
 .build();  
 client.newCall(request).enqueue(new Callback() {  
 @Override  
 public void onFailure(@NonNull Call call, @NonNull IOException e) {  
 e.printStackTrace();  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 Toast.*makeText*(MainActivity.this, "Conversion failed", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
 @Override  
 public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {  
 if (response.isSuccessful()) {  
 String responseBody = response.body().string();  
 String cleanedResponseBody = removeDuplicateBackslashes(responseBody);  
 Log.*d*("response", "respond" + responseBody);  
 Log.*d*("response", "respondclean" + cleanedResponseBody);  
 Gson gson = new Gson();  
 JsonElement jsonElement = gson.fromJson(responseBody, JsonElement.class);  
 if (jsonElement.isJsonObject()) { // Check if the response is a JSON object  
 JsonObject jsonObject = jsonElement.getAsJsonObject();  
 String result = jsonObject.get("result").getAsString();  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {Toast.*makeText*(MainActivity.this, "Server is Closed.", Toast.*LENGTH\_SHORT*).show();}});  
 } else {  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 String noStarString;  
 noStarString = cleanedResponseBody.replace("\*","");  
 String viewString;  
 viewString = noStarString.replace("\\operatorname{lim}","\\lim");  
 mathView.setTextAlignment(MTMathView.MTTextAlignment.*KMTTextAlignmentCenter*);  
 mathView.setFontSize(75f);  
 mathView.setLatex(viewString);  
 mathView.refreshDrawableState(); // CHANGING FONT BASED ON SIZE OF EQUATION  
 if(viewString.length()>80){  
 mathView.setFontSize(42f);  
 mathView.refreshDrawableState();  
 }  
 else if(viewString.length()>45){  
 mathView.setFontSize(55f);  
 mathView.refreshDrawableState();  
 }  
 if(mathView.getLatex()==""){  
 Toast.*makeText*(MainActivity.this, "Wasnot able to analyze.", Toast.*LENGTH\_SHORT*).show();}  
 convertLatexToMathML(noStarString); // SENDING LATEX TO MATHML }}); }  
 } else {  
 runOnUiThread(new Runnable() {@Override  
 public void run() {  
 Toast.*makeText*(MainActivity.this, "Conversion failed", Toast.*LENGTH\_SHORT*).show();}});}}});}

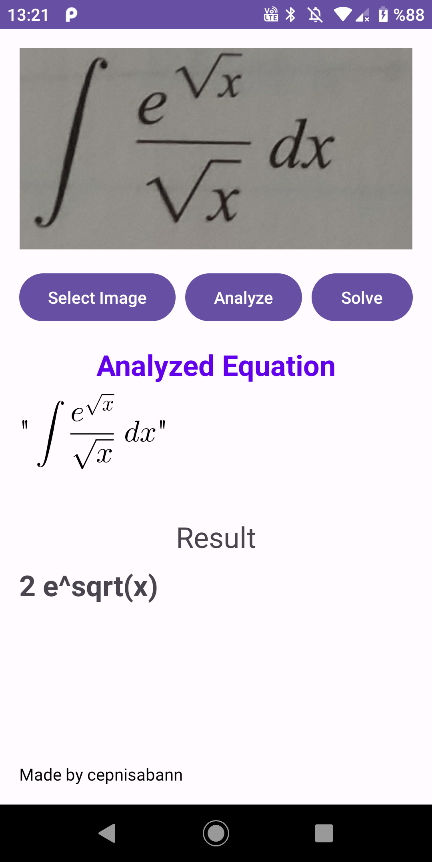
MathML and LaTex To Wolfram API [Code 2]

private void sendMathMLToWolfram(String mathMLString, boolean secondTry) {  
 //String plainText = StringEscapeUtils.unescapeXml(mathMLString);  
  
 OkHttpClient client = new OkHttpClient();  
 MediaType mediaType = MediaType.*parse*("application/x-www-form-urlencoded");  
 String wolframUrl = "https://api.wolframalpha.com/v2/result?appid=64XW9U-YV4Y2Q6U2V";  
  
 String postData = "input=" + Uri.*encode*(mathMLString);  
 RequestBody requestBody = RequestBody.*create*(mediaType, postData);  
  
 Request request = new Request.Builder()  
 .url(wolframUrl)  
 .post(requestBody)  
 .build();  
  
 client.newCall(request).enqueue(new Callback() {  
 @Override  
 public void onFailure(@NonNull Call call, @NonNull IOException e) {  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 Toast.*makeText*(MainActivity.this, "Request failed", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
  
 @Override  
 public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {  
 final String result = response.body().string();  
  
 if (response.isSuccessful()) {  
 // Parse the result to extract the relevant information  
  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 resultTextView.setText(result);  
 Log.*d*("response", "respond" + result);  
 if(secondTry){  
 resultTextView.setText("Result :\n"+result);  
 }  
 // Handle the equation result  
 }  
 });  
 } else {  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 if(checkOnce){  
 checkOnce= false;  
 resultTextView.setText("First method failed. Trying second method");  
 sendMathMLToWolfram(MathMLConverter.*fixMathML*(mathMLString),true);  
 }  
 Log.*d*("response", "respond" + result);  
 if(secondTry){  
 resultTextView.setText("Not me Wolfram says: "+ result);  
 if(result.contains("understand")){  
 resultTextView.setText("Wolfram did'nt understand your problem.");  
 }  
 }  
 }  
 });  
 }  
 }  
 });  
}

Latex to Mathml API code and fixing mathml [Code 3]

private void convertLatexToMathML(String latexString) {  
  
 OkHttpClient client = new OkHttpClient();  
 MediaType mediaType = MediaType.*parse*("application/json");  
 String texzillaUrl = "http://159.146.11.126:3000/convert";  
  
 // Create a JSON object with the "latex" key and the LaTeX string as its value  
 JsonObject jsonBody = new JsonObject();  
 jsonBody.addProperty("latex", latexString);  
  
 RequestBody requestBody = RequestBody.*create*(mediaType, jsonBody.toString());  
  
 Request request = new Request.Builder()  
 .url(texzillaUrl)  
 .post(requestBody)  
 .build();  
  
 client.newCall(request).enqueue(new Callback() {  
 @Override  
 public void onFailure(@NonNull Call call, @NonNull IOException e) {  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 Toast.*makeText*(MainActivity.this, "Conversion failed", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
  
 @Override  
 public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {  
 final String result = response.body().string();  
  
 if (response.isSuccessful()) {  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 wolframSend = result;  
 Log.*d*("convertLatexToMathML SEND", "result : " + result);  
 String newresult = MathMLConverter.*fixMathML*(result);  
 Log.*d*("convertLatexToMathML SEND", "result : " + newresult);  
  
 }  
 });  
 } else {  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
  
 Log.*d*("response", "respond" + result);  
  
 Toast.*makeText*(MainActivity.this, "Conversion failed", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
 }  
 });  
}

Analyzed And Solved Screen

****

All of my application codes are at https://github.com/cepnisabann/LatexPix

## REFERENCES

* 1. [Patel, A., Pandit, D., & Acharya, N. Android based Mathematical Expression Evaluation from Images.](https://www.ijert.org/research/android-based-mathematical-expression-evaluation-from-images-IJERTV5IS080475.pdf)
  2. [Sepetnitsky, V., & Dahan, Y. (Year). OCR of Math Expressions: Optical Character Recognition System of Mathematical Expressions and Testing Framework [Final project].](https://www.cs.bgu.ac.il/~ben-shahar/Teaching/Computational-Vision/StudentProjects/ICBV111/ICBV-2011-1-VitaliSepetnitsky-YakirDahan/index.php)
  3. [Polewczak, J. A. C. E. K. (2005). LaTeX, MathML, and TEX4HT: Tools for creating accessible documents. *Mathematics, CSUN, Northridge*, *24*.](https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=af663b9e9c133f9bd94012d5b5aeb4d0870ddc55)
  4. [Wolfram Alpha API Documentation](https://products.wolframalpha.com/api/documentation)
  5. [Optical Character Recognition For Mathematical Equations](file:///C:\Users\saban\Desktop\Stuff\1.%09https:\web.njit.edu\~avp38\projects\multi_projects\ocr.html)
  6. [Pix2Tex Documentation](https://pix2tex.readthedocs.io/en/latest/)