Title

Empirical Evaluation of Approaches for Digit Recognition
Abstract

Optical Character Recognition (OCR) is a well studied subject involving various application areas. OCR results in various limited problem areas are promising, however building highly accurate OCR application is still problematic in practice. This thesis discusses the problem of recognizing and confirming Bingo lottery numbers from a real lottery field, and a prototype for Android phone is implemented and evaluated. An OCR library Tesseract and two Artificial Neural Network (ANN) approaches are compared in an experiment and discussed. The results show that training a neural network for each number gives slightly higher results than Tesseract.

Keywords: Optical Character Recognition, OCR, Artificial Neural Network, ANN
Preface I enjoyed writing this thesis because OCR is a challenging problem to tackle for me and I got to use neural networks in a practical setting for the first time, which was exciting. The work was inspired by an earlier programming project where I struggled with an OCR problem. I would like to take the opportunity to thank my supervisors at the university.
Glossary

**ANN**  
Artificial Neural Network - Computational model of neurons in a brain, 1

**Google Goggles**  
OCR product of Google Inc., 1

**JavaCV**  
Java Computer Vision - Computer vision library with Java language front end, 1

**NDK**  
Android Native Developer Kit - A kit that allows native language (C++) programming on Android, 1

**OCR**  
Optical Character Recognition - The act of recognizing characters from an image, 1

**OpenCV**  
Open Computer Vision - Computer vision library, 1

**Tesseract**  
OCR programming library, 1

List of Figures

1.1 Bingo field ........................................... 1  
1.2 Recognition process .................................... 2  
2.3 Typical Feed Forward Network .......................... 7  
4.4 Main menu ............................................ 9  
4.5 Capture preview ........................................ 10  
4.6 Helper image .......................................... 10  
4.7 Recognition results .................................... 10  
4.8 Training feature ....................................... 11  
4.9 Settings .............................................. 11

List of Tables

1.2 Research Questions ..................................... 3  
5.3 Training image statistics ............................... 12  
5.4 Single network results .................................. 13  
5.5 Multiple network results ............................... 13
1 Introduction

First some background to the study is presented, then in Problem Definition the problem is motivated. Some background research and motivation is discussed in Previous Research, then specific research questions are formulated in Purpose and then after that limitations presented. Finally target group is recommended and an outline presented.

1.1 Introduction / Background

This thesis is about building an App for Android phones to recognize digits in a Bingo lottery field. Bingo is a type of lottery where the goal is to mark correct cells in a grid of drawn numbers. Victory condition is met if a whole row in the grid or four corners is filled. There are variations of the game with different conditions but in this thesis we only focus on this version. An example of a bingo field is shown in Figure 1.1.

![Figure 1.1: Bingo field](image)

The purpose of the App is to confirm that the field is correctly filled and if any victory condition is met. Since the app is about correcting human errors when manually playing Bingo, the accuracy of the application is of critical importance.

The problem consists of recognizing the lottery field which is made up by a square grid. The first step is recognizing the grid, which is done by finding the the size of connected component bounding box and then selecting the largest one. In that grid each cell contains a number with at most two digits. These numbers are segmented using connected component analysis. Connected component is a an area in an image that has pixels of same color adjacent to each other. Segmentation is done by selecting two largest connected components [1] [2].
Image recognition is a process, which usually consist of taking a picture, process the image then present the results and lastly go back to first step and correct if needed, as outlined in Figure 1.2. In order to increase the usability of the application some help methods have to be devised to help the user to correctly take a picture. The help methods need to detect if there were errors in recognition and notify the user if a new picture needs to be taken.

Figure 1.2: Recognition process

Historically Optical Character Recognition (OCR) research has focused on scanned documents, however the increase in mobile devices equipped with cameras has renewed interest in OCR. Application areas include image based searching, searching on words that appear in images, license plate recognition, puzzle solving, using images as input from receipts and invoices and handicap assistance.

The results of OCR tasks in limited problem areas are mostly 90% and upward [3] [4] [5]. Results show that it is possible to achieve promising results by using state of the art image processing and recognition frameworks such as OpenCV and Tesseract. Practical usability is problematic in applications that require 100% accuracy to function, therefore the problem studied is a challenging one.

1.2 Problem definition

There have been several attempts of comparing OCR engines, however these tests assume that input data is a document [6] [7]. Some tests are also quite old but recent results show 98% accuracy rate for Tesseract OCR engine [8]. This thesis compares an Open Source OCR engine and two Artificial Neural Network (ANN) approaches for a different
practical problem than document analysis. The problem presented is an OCR application that is intolerant against recognition errors. The challenge is building an OCR application for Android phones that reads numbers from a bingo lottery field.

1.3 Previous research

Matei, Pop and Valean used ANN together with K Nearest Neighbor to recognize license plates with 99.3% accuracy. Mutholib et al. used optimized template matching instead of ANNs and were able to achieve an accuracy of 97.46% for license plates. Because of these results an approach using ANN was included for comparison in this thesis. Casey and Lecolinet performed a study of character segmentation strategies, which provides an overview [9]. The approach used in this thesis is recognition based segmentation as proposed by Casey and Lecolinet. The segmentation approach used is also very common for document OCR where the connected components are extracted [10]. The idea to do some image processing in real time before taking the image is incorporated into the prototype. In this thesis some real-time connected component analysis is used on the image presented to the user [11].

1.4 Purpose and research questions

Mutholib et al. showed that optimized template matching is more accurate than ANNs for license plates [5]. Matei, Pop and Valean showed that ANNs together with K Nearest Neighbor achieves slightly higher accuracy for license plates. This raises Research Question 1 (RQ1), see Table 1.2. Tesseract uses several methods to preprocess and recognize letters [12]. To answer Research Question 1 (RQ1), this essay studies Tesseract and ANNs with the Backpropagation learning algorithm approach and compares them in an experiment. The comparison is useful as these approaches are the most popular choice for OCR applications.

PhotoOCR is an app developed by Google which has unknown techniques for text detection [10]. This gives reason to assume that there are several methods combined in Google Goggles. It is a Google product and little to no information is available about the inner workings. Google Goggles performs very well for Sudoku problem. This raises Research Question 1.1 (RQ1.1), see Table 1.2.

To answer Research Question 1.1 (RQ1.1), a prototype app for Android phone to recognize digits in a Bingo lottery field is built. The most accurate version of application produced by this thesis is compared against Google Goggles Sudoku recognition feature.

<table>
<thead>
<tr>
<th>RQ1</th>
<th>What is the most accurate library to use for doing OCR of digits: Tesseract or ANN?</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ1.1</td>
<td>How does our prototype compare to Google Goggles Sudoku recognition?</td>
</tr>
</tbody>
</table>

Table 1.2: Research Questions

1.5 Scope / Limitation

This thesis only compares Tesseract in single character mode and on digits. Full comparison is out of scope for this thesis. In this thesis classical Back Propagation training and Feed Forward neural networks are used. More advanced variations of networks and
training are not used. The resulting data provides some comparison of single network and multiple network approach. Full comparison of approaches to using ANNs is not the goal of this study and left for further research.

1.6 Target group

If the reader has an OCR problem and the problem is similar to the one presented in this thesis then this comparative study is relevant. The main purpose is to compare some popular choices. The reader should be aware that the source code of the prototype produced in this thesis is published as Open Source online [13] [14]. A goal of this thesis is that the open source code contributes a practical example and some extra knowledge to internet community.

1.7 Outline

- In the next chapter 2 background information is provided about Android, Tesseract and ANNs.
- Chapter 3 explains the scientific approach used in this thesis.
- Chapter 4 gives implementation details.
- Chapter 5 provides the results and data from experiments.
- Chapter 6 gives some analysis of data from experiments and adds some additional data.
- Chapter 7 discusses the results.
- Chapter 8 draws conclusions from the data.
2 Background / Theory

In this chapter we describe the Android operating system. Background to Various OCR and image processing libraries is presented. Furthermore ANNs and Back Propagation learning is explained shortly.

2.1 Android

This chapter explains different aspects of Android operating system relevant to this thesis.

2.1.1 Android background

Android is an operating system for mobile phones, tablets and possibly other devices. Android is based on Linux and is developed by Google Inc. Applications for Android are written in Java programming language and compiled to run in a Dalvik Virtual Machine (DVM). Dalvik is Open Source and is being replaced by Android Runtime (ART) in 4.4 Kitkat version of Android. Prototype of this thesis is run on a Samsung Smartphone using 4.1 JellyBean version of Android.

2.1.2 Android Software Development Kit

Developing for Android operating system is done with a Software Development Kit (SDK) provided by Google. The SDK includes a modified ready to use Eclipse environment. SDK also includes an Android emulator so that it is possible to develop without having access to a device running Android operating system. The emulator is able to use any web camera to serve as a camera in a mobile device. Using a web camera in an emulator has a drawback because there is a need to check if there is an auto-focus feature or not. The development of prototype used in this thesis is done for a real device, therefore results of using the prototype in an emulator and a web camera may differ.

2.1.3 Android Native Developer Kit

Android also has a Native Developer Kit (NDK). The NDK lets a developer compile code from native languages like C++ and then use that code with a Java interface. The use of NDK is discouraged as it increases the complexity of the code and a guideline is given: "In general, you should only use the NDK if it is essential to your app—never because you simply prefer to program in C/C++.” [15]. This thesis uses libraries that are compiled for NDK. To incorporate native libraries the NDK must be downloaded and appropriate make files must be used as input to the NDK tools.

2.2 Libraries and Artificial Neural Networks

In this chapter libraries are described. Also a brief explanation is given about ANNs.

2.2.1 JavaCV - OpenCV

JavaCV is a Java interface to the OpenCV computer vision library. OpenCV is an Open Source library with a large set of features. This thesis only uses a small set of functions documented in Imgproc [16]. OpenCV Manager is a library manager for Android, which is an automatic service. The purpose of the manager is to remove the need to include binary libraries of OpenCV together with your app. The manager installs one copy of the
library and all apps that are dependent on the library can use the same updated version [17].

2.2.2 Tesseract

Tesseract is an Open Source OCR library, which was originally developed by Hewlett Packard. "Tesseract is probably the most accurate open source OCR engine available. Combined with the Leptonica Image Processing Library it can read a wide variety of image formats and convert them to text in over 60 languages. It was one of the top 3 engines in the 1995 UNLV Accuracy test. Between 1995 and 2006 it had little work done on it, but since then it has been improved extensively by Google. It is released under the Apache License 2.0.” [18].

Tess-two is a fork of Tesseract tools for Android [19]. It provides a Java Interface to Tesseract library and an Android library project that makes it very easy to use in a project. Tess-two is used to build the prototype for this thesis.

2.3 Artificial Neural Networks

Artificial Neural Networks (ANNs) are biologically inspired models for computation of neurons in a mammal brain. The history of ANNs goes back to 1943 when McCulloch and Pitts showed a first computational model for neural networks [20]. ANNs is an area in computer science that is very popular for research. There are various models of ANNs and learning algorithms for different purposes. Usually a neural network is modelled by an array of computational units called Neurons. Each neuron has a number of weights that are numbers between -1 and +1. Neurons form a collection that is called a Layer. Collection of layers form a neural network. This thesis uses Feed Forward networks with the Backpropagation learning algorithm. These two choices are considered the most simple and most classical and are frequently explained using the problem of OCR [21].
2.3.1 Feed Forward Neural Networks

In a Feed Forward neural network output signal from neurons travel only in one direction towards the layer that is called the output layer. If any output signal from a neuron travels in a cycle the network is in the Recurrent neural network class, which is not used for the problem in this thesis. A typical model of a Feed Forward neural network is shown in Figure 2.3 [22].

![Figure 2.3: Typical Feed Forward Network](image)

2.3.2 Back Propagation Learning Algorithm

Back Propagation learning is done with examples. Input data is propagated through the network from input layer to output layer to get the output. The desired output is compared to the actual output and error is calculated. Error is used to calculate a new weight for a neuron to make the difference between actual and desired output smaller. The back propagation of the algorithm takes place when error is propagated backwards in the network. The error that back propagates is calculated by adding together the errors from previous neurons it is connected to.

The Backpropagation algorithm suffers from a problem called overfitting. Overfitting is when network is trained too much with training data and it loses it generalization capabilities on validation data. One simple solution against this problem is to initialize the network with different random weights and perform the training again.
3 Method

In this chapter the scientific approach for this thesis is explained and motivated. Prototype requirements are described. Some information about reliability is provided. Finally some information about the proper use of application is presented.

3.1 Scientific approach

In this thesis an experiment is done to compare Tesseract and ANNs. The comparison is done by collecting quantitative data about the accuracy of libraries. Data collection is done by the prototype application that takes a picture of a Bingo field and saves it. The picture can then be recognized or collected as training data. For Tesseract a number of images are taken and recognized by the prototype and error count is noted manually. The ANNs are compared using 10-fold cross validation. Finally a success and error ratio is presented as a percentage. Most OCR research uses success and error ratio as resulting data, therefore it was chosen for thesis as well. The working effort that it takes to use a library or ANN is not measured. The comparison of Google Goggles and the prototype is a small comparison of user experience not a statistical experiment.

3.1.1 Prototype requirements

Following requirements and features must be fulfilled for the prototype. Note that the prototype does not have to run in the emulated environment.

- Over 90% accuracy with at least one library
- Runs on a real device with Android version 4 or higher.
- Can be used to collect training data and Train ANNs
- Separate project for training ANNs.

3.2 Analysis

The data is collected through manual testing of the prototype. This thesis collects data from a controlled experiment and data is analyzed using statistical methods.

3.3 Reliability

Extracting the Bingo grid and thus the numbers from it reliably is problematic, therefore when collecting data the grid extraction errors are noted separately. When there is an error extracting the corners of the Bingo grid the recognition accuracy drops to nothing to do with accuracy of OCR library. The prototype was built for educational and experimental purpose. The confirmation of Bingo numbers should not be relied on and manual check is still encouraged.
4 Implementation

This chapter gives details about the ANN implementation and the prototype.

4.1 ANN implementation

The prototype can be configured to use single and multiple network approach. The single network uses 10 output nodes, one for each number. The multiple network approach has a 10 networks with a single output node. Multiple networks are trained with a separate copy of the training data for each of its networks. All pictures of the number a particular network is supposed to recognize must return an output of 1 and all other pictures 0. For example to train network that recognizes number 3 the training data is copied and the target output is set to 1 for all training images that contain number 3 and target is 0 for all other images. If training data contains 1946 images then a copy is made of it 10 times for each network making it a total of 10*1946 training instances. Recognition in single network is done by propagating the image pixel values through the network and choosing the highest value in the output array that is over 0.5 threshold. If none are above the threshold then the image is not a number. In multiple network approach the image is propagated through 10 networks. Each network output is put into an array. Then same function as for single network (choosing largest output that is higher than threshold) is used to read the output value.

4.2 Main menu

[Figure 4.4: Main menu]

Figure 4.4 shows the main menu of the prototype. Main menu has 4 buttons. Capture starts the camera to take a picture of a bingo grid. Enter numbers is for entering correct numbers of bingo. Train opens up a way to collect training data and train networks. Test train is for testing training.

4.3 Capture

[Figure 4.5 is the camera preview. It is pointed to a bingo ticket. Besides the preview there are 3 buttons. Capture button takes a picture. Recognize is pressed when a black and white bingo grid appears after capturing. New button is to start the preview again and take another attempt at capturing.]
4.4 Helper image

Figure 4.6 is the helper image presented to the user before actually recognizing any numbers. It should not be confused with recognizing numbers because at this stage the numbers have not been run through Tesseract or ANNs. In this picture the grid is recognized properly and the user should press Recognize button.

4.5 Recognition

Figure 4.7 is a grid where numbers have been recognized. There are errors in it. There are numbers in bold, which are entered from Enter Numbers from main menu. These are the numbers that are being confirmed at the moment.

4.6 Training and data collection

Figure 4.8 is the feature that is used to collect training data and train ANNs. Because mobile phone has less resources then a laptop there is a separate project for training ANNs. To collect data one sees an image from a previous capture and enters the number it sees then presses Next to get the next number.
4.7 Settings

Figure 4.9 show the settings for the prototype. Here one can choose between Tesseract, single network and multiple network to be used during recognition.
5 Results / Empirical data

This chapter contains data collected from the experiment. Chapter 5.1 outlines the experiment. The following chapters present results of the experiment for different recognition approaches.

5.1 ANN comparison

To compare the single and multiple network approach a 10-fold cross validation is done. The source code together with training data is available online [13] [14]. The training data has 2162 training images. For each iteration 10 percent of data (216 images) is used for validation and remaining 90 percent (1946 images) for training. It should be noted that multiple network approach makes a separate copy of the training set for each network, increasing the actual size of training data. See chapter 4.1 for details. The training data was collected manually using the prototype application from real bingo grids. A picture was taken of a real grid, then number were manually recognized. The 10 fold cross validation was automatically run in a separate java project. Both single and multiple network approach is trained to either 99 percent accuracy on the training set or maximum of 1000 iterations of training. Input data to ANNs is 18x18=324 pixel binary image where the number can be at any position. Table 5.3 contains statistics about the training data that was collected and used for training. Note that training data contained images that were empty or contained some noise that is not any number.

<table>
<thead>
<tr>
<th>Number</th>
<th>Image count in training data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>264</td>
</tr>
<tr>
<td>2</td>
<td>216</td>
</tr>
<tr>
<td>3</td>
<td>243</td>
</tr>
<tr>
<td>4</td>
<td>233</td>
</tr>
<tr>
<td>5</td>
<td>258</td>
</tr>
<tr>
<td>6</td>
<td>255</td>
</tr>
<tr>
<td>7</td>
<td>183</td>
</tr>
<tr>
<td>8</td>
<td>95</td>
</tr>
<tr>
<td>9</td>
<td>103</td>
</tr>
<tr>
<td>Noise/Empty</td>
<td>207</td>
</tr>
</tbody>
</table>

Table 5.3: Training image statistics
5.2 Single network results

Single network has a setup of 324 input nodes, 60 hidden nodes and 10 output nodes. Table 5.4 shows results for single network training and validation.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Accuracy on training set (%)</th>
<th>Accuracy on validation set (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>93.83</td>
<td>83.79</td>
</tr>
<tr>
<td>2</td>
<td>93.06</td>
<td>54.62</td>
</tr>
<tr>
<td>3</td>
<td>94.34</td>
<td>84.25</td>
</tr>
<tr>
<td>4</td>
<td>94.39</td>
<td>86.57</td>
</tr>
<tr>
<td>5</td>
<td>93.11</td>
<td>80.09</td>
</tr>
<tr>
<td>6</td>
<td>93.26</td>
<td>83.79</td>
</tr>
<tr>
<td>7</td>
<td>94.09</td>
<td>70.83</td>
</tr>
<tr>
<td>8</td>
<td>94.03</td>
<td>90.74</td>
</tr>
<tr>
<td>9</td>
<td>93.01</td>
<td>80.09</td>
</tr>
<tr>
<td>10</td>
<td>95.11</td>
<td>63.42</td>
</tr>
<tr>
<td><strong>Arithmetic mean</strong></td>
<td><strong>93.82</strong></td>
<td><strong>77.81</strong></td>
</tr>
</tbody>
</table>

5.3 Multiple network results

Multiple network setup has 10 networks. Each network recognizes one number. Each network has a setup of 324 input nodes, 10 hidden nodes and 1 output node. The results are percentages for 10 networks combined. Individual network percentages are not included. Chapter 4.1 gives more detailed description of implementation.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Accuracy on training set (%)</th>
<th>Accuracy on validation set (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>99.30</td>
<td>97.31</td>
</tr>
<tr>
<td>2</td>
<td>99.51</td>
<td>93.56</td>
</tr>
<tr>
<td>3</td>
<td>99.46</td>
<td>98.37</td>
</tr>
<tr>
<td>4</td>
<td>98.63</td>
<td>97.73</td>
</tr>
<tr>
<td>5</td>
<td>99.28</td>
<td>97.82</td>
</tr>
<tr>
<td>6</td>
<td>99.36</td>
<td>98.19</td>
</tr>
<tr>
<td>7</td>
<td>99.50</td>
<td>95.04</td>
</tr>
<tr>
<td>8</td>
<td>99.48</td>
<td>98.42</td>
</tr>
<tr>
<td>9</td>
<td>99.45</td>
<td>97.31</td>
</tr>
<tr>
<td>10</td>
<td>99.31</td>
<td>94.62</td>
</tr>
<tr>
<td><strong>Arithmetic mean</strong></td>
<td><strong>99.33</strong></td>
<td><strong>96.84</strong></td>
</tr>
</tbody>
</table>

5.4 Tesseract results

Tesseract is not tested with 10-fold cross validation. Only a single test is performed. Tesseract was tested on 236 images captured by the prototype. It correctly identified 227 numbers and had 9 errors. This gives total accuracy of 96.18% for Tesseract. Most errors were recognizing 0 as a 6 or 9. Note that faulty pictures parsed by the prototype caused more errors, but these were not included in the test. Only Tesseract results are presented.
5.5 Google Goggles and Prototype test

20 pictures of different Sudokus were taken with Google Goggles. Out of 20 attempts 16 times it correctly identified the Sudoku grid and recognized 100% of its numbers. 20 pictures of different bingo grids were taken with the prototype. Out of 20 attempts 7 times it correctly identified the bingo grid.
6 Analysis

Results for single network, multiple network and Tesseract are 77.81\%, 96.84\% and 96.18\% respectively. It should be noted that accuracy on training set is considerably lower for single network then multiple network. There is a difference between single network approach and multiple network approach in how they use training data. See chapter 4.1 for details. Training time was not included in the experiment but both networks had at most 1000 iterations for training. Some extra tests were made with single network configuration 324, 20, 10 and 324, 40, 10 and 1000 iterations but the resulting accuracy was even lower. With single network configuration 324, 100, 10 and 2500 training iterations the results improved 3\% - 5\%. The errors of grid recognition are not included any of the results, which means that practical results of the prototype are lower. Usually there is need to take several pictures of the same grid.
7 Discussion

This chapter discusses the results obtained from the experiment and suggests improvements for the prototype. Some thoughts about the experiment are included in the end of the chapter.

7.1 Problem solving / Results

The results of Tesseract might be improved if images of correct DPI are provided. The results of ANNs can probably be improved if the number in the training and recognition image is moved to one corner or always centered. It should be noted that ANNs were trained specifically with data and images of problem presented by this thesis whereas Tesseract is a more general engine.

First attempt at making helper methods described in the Introduction was unsuccessful. The first attempt consisted of manipulating the Android camera buffer before the preview to draw a rectangle around the bingo grid. The frame rate of preview dropped very low and the rectangle did not stay fixed but moved around from frame to frame. OpenCV library however has documented support for Android camera manipulation. With extra work it is possible to achieve this. The final solution is simpler. The user is presented with a cut out black and white bingo grid if the picture was taken correctly. Then user can then proceed to recognition phase.

7.2 Method reflection

Tesseract was not tested 10 times with different numbers to get an average accuracy. This makes the small difference in final result less relevant. The bingo tickets changed during the work on this thesis so the tickets acquired later had more fields and less spacing between some numbers. That means there are numbers that are connected in an image and they need to be segmented. The implementation difference in single and multiple network mean that they use different number of training images total in practice.
8 Conclusion

This thesis compared the Tesseract and two different neural network approaches for the purpose of optical character recognition. A single test for Tesseract and 10-fold cross validation was done on the ANNs. A prototype was built for Android that can recognize numbers from a bingo lottery. A small comparison between Google Goggles and the prototype was performed.

8.1 Conclusions

This section gives short answers to the research questions.

- **RQ1**: What is the most accurate library to use for doing OCR of digits: Tesseract or ANN?
- **RQ1.1**: How does our prototype compare to Google Goggles Sudoku recognition?

8.1.1 Research Question 1 (RQ1)

The results show that it is possible to achieve same accuracy with ANNs then Tesseract, however the difference in results is not significant. It took about 2 days to collect the training data for ANN approach and another 6 hours to train. It is a personal choice weather to use Tesseract or ANNs, however when using ANNs the multiple network approach is a recommended choice.

8.1.2 Research Question 1.1 (RQ1.1)

The results show that total recognition rate for the prototype (35%) is lower then Google Goggles (80%).

8.2 Further research

The resulting data indicates that multiple network approach performs better then single network approach. This comparison was not the goal of the study. There is not enough data to say if single network is slower to train or less accurate. To see if multiple networks perform better generally could be an area of further research.

The ANN approach is not a general solution. The ANN approach is specifically trained for the problem presented for this study. Tesseract is a general OCR engine, which performs well. To study if ANN based approach could be used as a general solution might be an area of study.

The help method in the final prototype of this study is a simple one. To draw a bounding box in real time in camera preview is possible but requires more work. OpenCV has documented support for camera preview in Android. It could be an area of further study to see how to accomplish this with high frame rate.
References


[17] “Opencv manager,”


[22] “Neural networks diagram,”